驱动基础单元kobject操作与测试

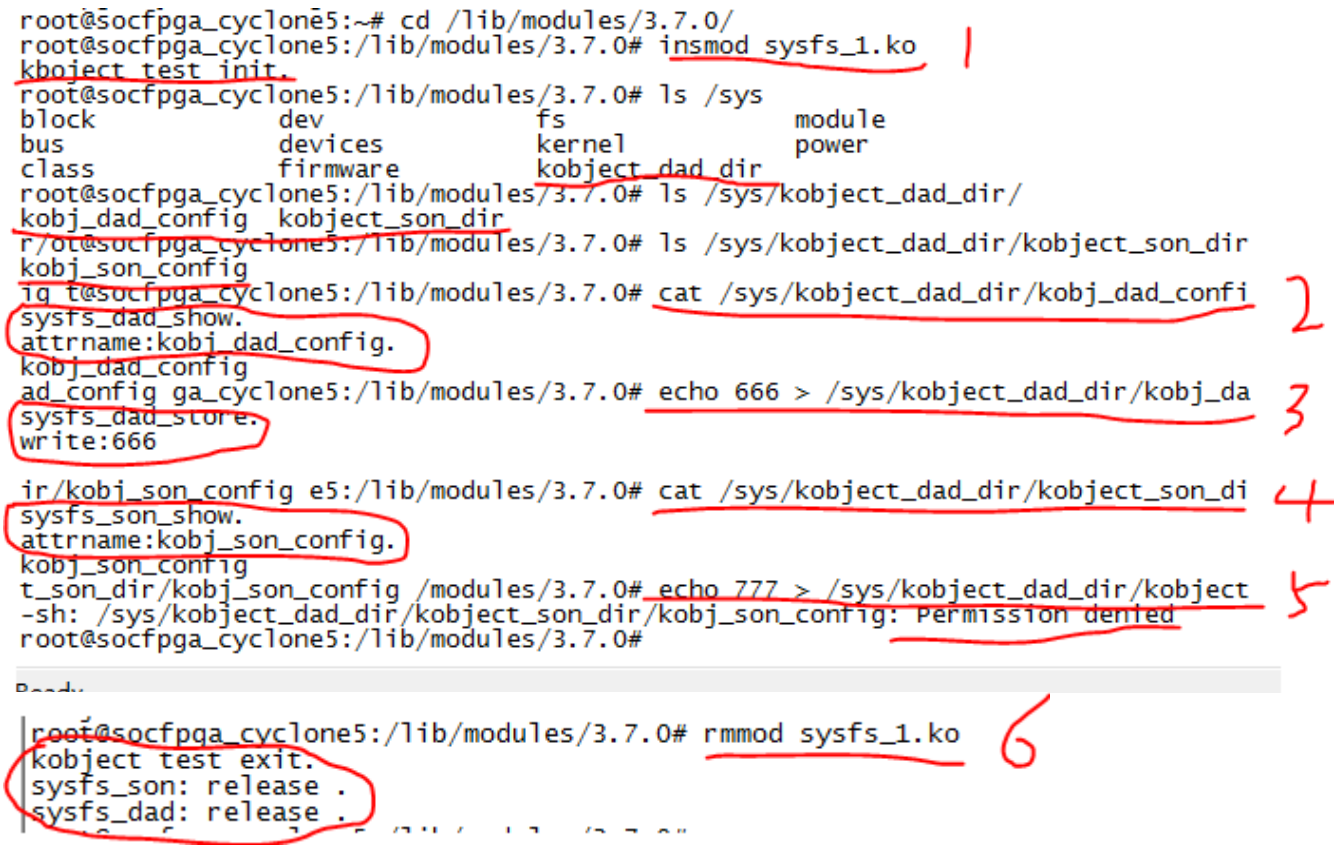
1.代码

```
#include <linux/device.h>
#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/string.h>
#include <linux/sysfs.h>
#include <linux/stat.h>
#include <linux/slab.h>
#include <linux/uaccess.h>

struct kobject *kobj_dad;
struct kobject *kobj_son;

void obj_dad_release(struct kobject *kobject);
ssize_t kobj_dad_show(struct kobject *kobject, struct attribute *attr,char *buf);
ssize_t kobj_dad_store(struct kobject *kobject,struct attribute *attr,const char *buf,
                       size_t count);
void obj_son_release(struct kobject *kobject);
ssize_t kobj_son_show(struct kobject *kobject, struct attribute *attr,char *buf);
ssize_t kobj_son_store(struct kobject *kobject,struct attribute *attr,const char *buf,
                       size_t count);

/* 文件名和权限 */
struct attribute dad_attr = {
    .name = "kobj_dad_config",
    .mode = S_IRWXUGO,
};
struct attribute son_attr = {
    .name = "kobj_son_config",
    .mode = S_IRUGO,
};

/* 这里是数组,即目录下可以有多个文件 */
static struct attribute *def_dad_attrs[] = {
    &dad_attr, /* 目录下的一个文件 */
    NULL,
};
static struct attribute *def_son_attrs[] = {
    &son_attr, /* 目录下的一个文件 */
    NULL,
};

/* 属性的读写 */
struct sysfs_ops obj_dad_sysops = {
     .show = kobj_dad_show,
```

```
     .store = kobj_dad_store,
};
struct sysfs_ops obj_son_sysops = {
     .show = kobj_son_show,
     .store = kobj_son_store,
};

/* 属性和操作集合 */
struct kobj_type ktype_dad = {
    .release = obj_dad_release,
    .sysfs_ops = &obj_dad_sysops,
    .default_attrs = def_dad_attrs,
};
struct kobj_type ktype_son = {
    .release = obj_son_release,
    .sysfs_ops = &obj_son_sysops,
    .default_attrs = def_son_attrs,
};

/* 释放kobject时候调用的回调函数 */
void obj_dad_release(struct kobject *kobject)
{
    printk("sysfs_dad: release .\n");
    kfree(kobject);
}
void obj_son_release(struct kobject *kobject)
{
    printk("sysfs_son: release .\n");
    kfree(kobject);
}

/* 读属性时调用的函数 */
ssize_t kobj_dad_show(struct kobject *kobject, struct attribute *attr,char *buf)
{
    printk("sysfs_dad_show.\n");
    printk("attrname:%s.\n",attr->name);
    sprintf(buf,"%s\n",attr->name);
    return strlen(attr->name) + 2;
}
ssize_t kobj_son_show(struct kobject *kobject, struct attribute *attr,char *buf)
{
    printk("sysfs_son_show.\n");
    printk("attrname:%s.\n",attr->name);
    sprintf(buf,"%s\n",attr->name);
    return strlen(attr->name) + 2;
}

/* 写属性时调用的函数 */
ssize_t kobj_dad_store(struct kobject *kobject,struct attribute *attr,
                       const char *buf, size_t count)
{
    printk("sysfs_dad_store.\n");
    printk("write:%s\n",buf);
```

```
    return count;
}
ssize_t kobj_son_store(struct kobject *kobject,struct attribute *attr,
                       const char *buf, size_t count)
{
    printk("sysfs_son_store.\n");
    printk("write:%s\n",buf);
    return count;
}

/* 加载模块时调用 */
static int __init kobj_test_init()
{
    printk("kboject test init.\n");
    kobj_dad = kzalloc(sizeof(struct kobject), GFP_KERNEL);           /* 初始化kobject和注册进内核前要清零 */
    kobject_init_and_add(kobj_dad,&ktype_dad,NULL,"kobject_dad_dir"); /* 创建一个目录,kobj引用计数加1，注册进内核,parent为NULL,即在跟目录下 */

    kobj_son = kzalloc(sizeof(struct kobject), GFP_KERNEL);
    kobject_init_and_add(kobj_son,&ktype_son,kobj_dad,"kobject_son_dir"); /* 创建一个目录,kobj_son引用计数加1，kobj_dad引用计数加1,注册进内核 */

    return 0;
}

/* 卸载模块时调用 */
static int __exit kobj_test_exit()
{
    printk("kobject test exit.\n");
    kobject_del(kobj_son);  /* 删除kobj_son目录,kobj_dad引用数减1,kobj_son没有减 */
    kobject_put(kobj_son);  /* kobj_son引用数减1 */
    kobject_del(kobj_dad);
    kobject_put(kobj_dad);

    return 0;
}

module_init(kobj_test_init);
module_exit(kobj_test_exit);

MODULE_AUTHOR("Yeshen 569242715@qq.com");
MODULE_LICENSE("GPL v2");
```

2.简要说明



Kobject的核心功能是：保持一个引用计数，当该计数减为0时，自动释放Kobject所占用的meomry空间。操作过程如上图所示：

1.insmod sysfs_1.ko 加载上面代码编译得到的模块sysfs_1.ko，然后打印出kboject test init.，因此运行了kobj_test_init()函数。

2.cat /sys/kobject_dad_dir/kobj_dad_config 通过上面命令读对应/sys/kobject_dad_dir/kobj_dad_config的属性，打印sysfs_dad_show.和attrname:kobj_dad_config.，因此运行了kobj_dad_show函数。

3.echo 666 > /sys/kobject_dad_dir/kobj_dad_config 通过上面命令写666到/sys/kobject_dad_dir/kobj_dad_config，因为该属性是可写的，所以写该属性会调用kobj_dad_store，打印了sysfs_dad_store.和write:666。

4.cat /sys/kobject_dad_dir/kobj_son_dir/kobj_son_config 通过上面命令读对应/sys/kobject_dad_dir/kobj_son_dir/kobj_son_config的属性，打印sysfs_son_show.和attrname:kobj_son_config.，因此运行了kobj_son_show函数。

5.echo 777 > /sys/kobject_dad_dir/kobj_son_dir/kobj_son_config 通过上面命令写777到/sys/kobject_dad_dir/kobj_son_dir/kobj_son_config，因为写该属性是不可写的，因此不调用kobj_son_store，并且内核打印了 Permission denied。

6.rmmod sysfs_1.ko 卸载sysfs_1.ko模块，会首先调用kobj_test_exit，因此会先打印kobject test exit.，然后通过kobject_put(kobj_son)减少kobj_son的引用计数，变为0，因此会回调ktype_son中的realse函数，即obj_son_release，因此接着打印sysfs_son: release .，最后kobject_put(kobj_dad)将kobj_dad的引用计数减为0，调用obj_dad_release打印sysfs_dad: release .