```
/* ANRC RHKI */
/* Lab16: KObject Example */
#include <linux/kobject.h>
#include <linux/string.h>
#include <linux/sysfs.h>
#include <linux/module.h>
#include <linux/init.h>

#define DRIVER_AUTHOR   "ANRC"
#define DRIVER_DESC     "Lab16"

MODULE_LICENSE("GPL");           // Get rid of taint message by declaring code as GPL.

/*  Or with defines, like this: */
MODULE_AUTHOR(DRIVER_AUTHOR);    // Who wrote this module?
MODULE_DESCRIPTION(DRIVER_DESC); // What does this module do?

static int anrc_int1;
static int anrc_int2;

static ssize_t anrc_read_int(struct kobject *kobj, struct kobj_attribute *attr, char *buf)
{
        int var;

        if (strcmp(attr->attr.name, "anrc_int1") == 0)
                var = anrc_int1;
        else
                var = anrc_int2;
        return sprintf(buf, "%d\n", var);
}

static ssize_t anrc_write_int(struct kobject *kobj, struct kobj_attribute *attr, const char *buf, size_t
count)
{
        int var;

        sscanf(buf, "%du", &var);
        if (strcmp(attr->attr.name, "anrc_int1") == 0)
                anrc_int1 = var;
        else
                anrc_int2 = var;

        return count;
}

static struct kobj_attribute anrc_int1_attribute = __ATTR(anrc_int1, 0666, anrc_read_int, anrc_write_int);
static struct kobj_attribute anrc_int2_attribute = __ATTR(anrc_int2, 0666, anrc_read_int, anrc_write_int);

/* Create a group of attributes so that we can create and destroy them at the same time. */

static struct attribute *attrs[] = {
        &anrc_int1_attribute.attr,
        &anrc_int2_attribute.attr,
        NULL,   /* need to NULL terminate the list of attributes */
};

/*
 * An unnamed attribute group will put all of the attributes directly in
 * the kobject directory.  If we specify a name, a subdirectory will be
 * created for the attributes with the directory being the name of the
 * attribute group.
 */
static struct attribute_group attr_group = {
        .attrs = attrs,
};

static struct kobject *example_kobj;
```

```c
int init(void);
void cleanup(void);

int init(void)
{
        int retval;

        /*
         * Create a simple kobject with the name of "kobject_example",
         * located under /sys/kernel/
         *
         * As this is a simple directory, no uevent will be sent to
         * userspace.  That is why this function should not be used for
         * any type of dynamic kobjects, where the name and number are
         * not known ahead of time.
         */
        example_kobj = kobject_create_and_add("kobject_example", kernel_kobj);
        if (!example_kobj)
                return -ENOMEM;

        /* Create the files associated with this kobject */
        retval = sysfs_create_group(example_kobj, &attr_group);
        if (retval)
                kobject_put(example_kobj);

        return retval;
}

void cleanup(void)
{
        kobject_put(example_kobj);
}

module_init(init);
module_exit(cleanup);
```