Paleti Krishnasai
CED18I039

Device Drivers Lab 6

Driver code

```
#include<linux/kernel.h>
#include<linux/init.h>
#include<linux/module.h>
#include<linux/kdev_t.h>
#include<linux/fs.h>
#include<linux/cdev.h>
#include<linux/device.h>
#include<linux/slab.h>
#include<linux/uaccess.h>

#define mem_size 1024

dev_t dev = 0;
static struct class *dev_class;
static struct cdev my_cdev;
uint8_t *kernel_buffer;

static int __init chr_driver_init(void);
static void __exit chr_driver_exit(void);
static int my_open(struct inode *inode, struct file *file);
static int my_release(struct inode *inode, struct file *file);
static ssize_t my_read(struct file *filp, char __user *buf, size_t len,
loff_t *off);
static ssize_t my_write(struct file *filp, const char *buf, size_t len,
loff_t *off);


static struct file_operations fops =
{
   .owner = THIS_MODULE,
   .read = my_read,
   .write = my_write,
   .open = my_open,
   .release = my_release,
};
```

```c
static int my_open(struct inode *inode, struct file *file)
{
    /* creating physical memory */
    if((kernel_buffer = kmalloc(mem_size, GFP_KERNEL)) == 0) {
        printk(KERN_INFO"Cannot allocate the memory to the kernel..\n");
        return -1;
    }
    printk(KERN_INFO"Device File opened...\n");
    return 0;
}

static int my_release(struct inode *inode, struct file *file)
{
    kfree(kernel_buffer);
    printk(KERN_INFO"Device File closed...\n");
    return 0;
}

static ssize_t my_read(struct file *filp, char __user *buf, size_t len,
loff_t *off)
{
    copy_to_user(buf, kernel_buffer, mem_size);
    printk(KERN_INFO"Data read: DONE...\n");
    return mem_size;
}

static ssize_t my_write(struct file *filp, const char __user *buf, size_t
len, loff_t *off)
{
    copy_from_user(kernel_buffer, buf, len);
    printk(KERN_INFO"Data is written successfully...\n");
    return len;
}

static int __init chr_driver_init(void)
{
    /* Allocating Major number */
    if((alloc_chrdev_region(&dev,0,1,"my_dev"))<0) {
        printk(KERN_INFO"Cannot allocate the major number..\n");
        return -1;
    }

    printk(KERN_INFO"Major = %d Minor = %d..\n", MAJOR(dev), MINOR(dev));

    /* creating cdev structure */
    cdev_init(&my_cdev, &fops);

    /* Adding character device to the system */
    if((cdev_add(&my_cdev, dev, 1)) < 0) {
        printk(KERN_INFO"Cannot add the device to the system..\n");
        goto r_class;
    }

    /* creating struct class */
    if((dev_class = class_create(THIS_MODULE,"my_class")) == NULL) {
        printk(KERN_INFO"cannot create the struct class...\n");
        goto r_class;
    }

    /* creating device */
    if((device_create(dev_class,NULL,dev,NULL,"my_device"))==NULL) {
        printk(KERN_INFO"cannot create the device..\n");
        goto r_device;
    }

    printk(KERN_INFO"Device driver insert...done properly...\n");
    return 0;

r_device:
    class_destroy(dev_class);

r_class:
    unregister_chrdev_region(dev, 1);
    return -1;
}

void __exit chr_driver_exit(void) {
    device_destroy(dev_class, dev);
    class_destroy(dev_class);
```

```
   cdev_del(&my_cdev);
   unregister_chrdev_region(dev, 1);
   printk(KERN_INFO"Device driver is removed successfully..\n");
}

module_init(chr_driver_init);
module_exit(chr_driver_exit);

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("The character device driver");
```

Test file

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<unistd.h>

int8_t write_buf[1024];
int8_t read_buf[1024];

int main()
{
   int fd;
   int option;

   printf("Welcome to the demo of character device driver...\n");

   fd = open("/dev/my_device",O_RDWR);
   if(fd < 0) {
       printf("Cannot open device file...\n");
       return 0;
   }

   while(1) {
       printf("***please enter your option***");
```

```
        printf("          1. Write        ");
        printf("          2. Read         ");
        printf("          3. Exit         ");
        scanf("%d", &option);
        printf(" your option = %d\n",option);

        switch(option) {
            case 1:
                printf("Enter the string to write into the driver:\n");
                scanf("%s", write_buf);
                printf("Data written...");
                write(fd, write_buf, strlen(write_buf)+1);
                printf("DONE...\n");
                break;
            case 2:
                printf("Data is reading...");
                read(fd, read_buf, 1024);
                printf("Done...\n\n");
                printf("Data = %s\n\n", read_buf);
                break;
            case 3:
                close(fd);
                exit(1);
                break;
            default:
                printf("Enter valid option = %c\n", option);
                break;
        }
    }
    close(fd);
}
```

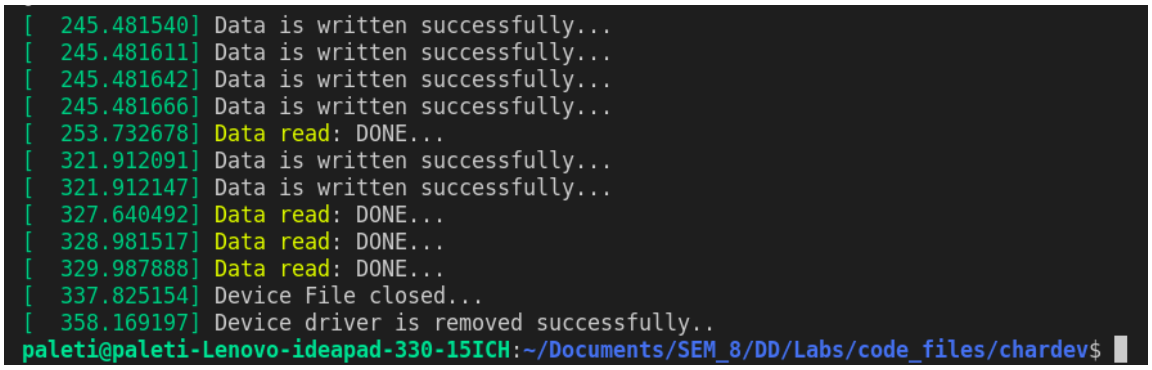
Makefile

```
# To build modules outside of the kernel tree, we run "make"
# in the kernel source tree; the Makefile these then includes this
# Makefile once again.
# This conditional selects whether we are being included from the
# kernel Makefile or not.
```

Paleti Krishnasai
CED18I039

```
LDDINC=$(PWD)/../include
EXTRA_CFLAGS += -I$(LDDINC)

ifeq ($(KERNELRELEASE),)

   # Assume the source tree is where the running kernel was built
   # You should set KERNELDIR in the environment if it's elsewhere
   KERNELDIR ?= /lib/modules/$(shell uname -r)/build
   # The current directory is passed to sub-makes as argument
   PWD := $(shell pwd)

modules:
   $(MAKE) -C $(KERNELDIR) M=$(PWD) modules

modules_install:
   $(MAKE) -C $(KERNELDIR) M=$(PWD) modules_install

clean:
   rm -rf *.o *~ core .depend .*.cmd *.ko *.mod.c .tmp_versions *.mod
modules.order *.symvers

.PHONY: modules modules_install clean

else
   # called from kernel build system: just declare what our modules are
   obj-m := cd.o
endif
```

Output:



```
[  245.481540] Data is written successfully...
[  245.481611] Data is written successfully...
[  245.481642] Data is written successfully...
[  245.481666] Data is written successfully...
[  253.732678] Data read: DONE...
[  321.912091] Data is written successfully...
[  321.912147] Data is written successfully...
[  327.640492] Data read: DONE...
[  328.981517] Data read: DONE...
[  329.987888] Data read: DONE...
[  337.825154] Device File closed...
[  358.169197] Device driver is removed successfully..
paleti@paleti-Lenovo-ideapad-330-15ICH:~/Documents/SEM_8/DD/Labs/code_files/chardev$ 
```