Device Drivers

Github Link : https://github.com/prateekagrawaliiit/Device-Drivers

Name :	Prateek Agrawal
Roll Number :	CED18I040
Date :	10th April 2022

Lab Exercise 6:

Write a C program. Compile. Insert the compiled program as a Kernel Module in Kernel. Remove the module from Kernel. Verify whether the insertion and deletion happened properly.

Lab Exercise 6 - Write a simple Char Device Driver Program in C language. Compile it as a kernel module. Insert in the kernel. Check whether the device file is created in proper directory. Check by writing data into the device file and reading data from the device file.

Code

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

#define MAX_DEV 1
```

```
static int mychardev_open(struct inode *inode, struct file *file);
static int mychardev_release(struct inode *inode, struct file *file);
static long mychardev_ioctl(struct file *file, unsigned int cmd,
unsigned long arg);
static ssize_t mychardev_read(struct file *file, char __user *buf,
size_t count, loff_t *offset);
static ssize_t mychardev_write(struct file *file, const char __user
*buf, size_t count, loff_t *offset);

static const struct file_operations mychardev_fops = {
   .owner      = THIS_MODULE,
   .open       = mychardev_open,
   .release    = mychardev_release,
   .unlocked_ioctl = mychardev_ioctl,
   .read       = mychardev_read,
   .write       = mychardev_write
};

struct mychar_device_data {
   struct cdev cdev;
};

static int dev_major = 0;
static struct class *mychardev_class = NULL;
static struct mychar_device_data mychardev_data[MAX_DEV];

static int mychardev_uevent(struct device *dev, struct
kobj_uevent_env *env)
{
   add_uevent_var(env, "DEVMODE=%#o", 0666);
   return 0;
}

static int __init mychardev_init(void)
{
   int err, i;
   dev_t dev;

   err = alloc_chrdev_region(&dev, 0, MAX_DEV, "mychardev");
```

```
    dev_major = MAJOR(dev);

    mychardev_class = class_create(THIS_MODULE, "mychardev");
    mychardev_class->dev_uevent = mychardev_uevent;

    for (i = 0; i < MAX_DEV; i++) {
      cdev_init(&mychardev_data[i].cdev, &mychardev_fops);
      mychardev_data[i].cdev.owner = THIS_MODULE;

      cdev_add(&mychardev_data[i].cdev, MKDEV(dev_major, i), 1);

      device_create(mychardev_class, NULL, MKDEV(dev_major, i), NULL,
"mychardev-%d", i);
    }

    return 0;
}

static void __exit mychardev_exit(void)
{
    int i;

    for (i = 0; i < MAX_DEV; i++) {
      device_destroy(mychardev_class, MKDEV(dev_major, i));
    }

    class_unregister(mychardev_class);
    class_destroy(mychardev_class);

    unregister_chrdev_region(MKDEV(dev_major, 0), MINORMASK);
}

static int mychardev_open(struct inode *inode, struct file *file)
{
    printk("MYCHARDEV: Device open\n");
    return 0;
}
```

```
static int mychardev_release(struct inode *inode, struct file *file)
{
   printk("MYCHARDEV: Device close\n");
   return 0;
}

static long mychardev_ioctl(struct file *file, unsigned int cmd,
unsigned long arg)
{
   printk("MYCHARDEV: Device ioctl\n");
   return 0;
}

static ssize_t mychardev_read(struct file *file, char __user *buf,
size_t count, loff_t *offset)
{
   uint8_t *data = "Hello from the kernel world!\n";
   size_t datalen = strlen(data);

   printk("Reading device: %d\n",
MINOR(file->f_path.dentry->d_inode->i_rdev));

   if (count > datalen) {
     count = datalen;
  }

  if (copy_to_user(buf, data, count)) {
     return -EFAULT;
  }

  return count;
}

static ssize_t mychardev_write(struct file *file, const char __user
*buf, size_t count, loff_t *offset)
{
   size_t maxdatalen = 30, ncopied;
   uint8_t databuf[maxdatalen];
```

```
   printk("Writing device: %d\n",
MINOR(file->f_path.dentry->d_inode->i_rdev));

   if (count < maxdatalen) {
     maxdatalen = count;
  }

  ncopied = copy_from_user(databuf, buf, maxdatalen);

  if (ncopied == 0) {
     printk("Copied %zd bytes from the user\n", maxdatalen);
  } else {
     printk("Could't copy %zd bytes from the user\n", ncopied);
  }

  databuf[maxdatalen] = 0;

  printk("Data from the user: %s\n", databuf);

  return count;
}
MODULE_LICENSE("GPL");
MODULE_AUTHOR("Prateek Agrawal - CED18I040");

module_init(mychardev_init);
module_exit(mychardev_exit);
```

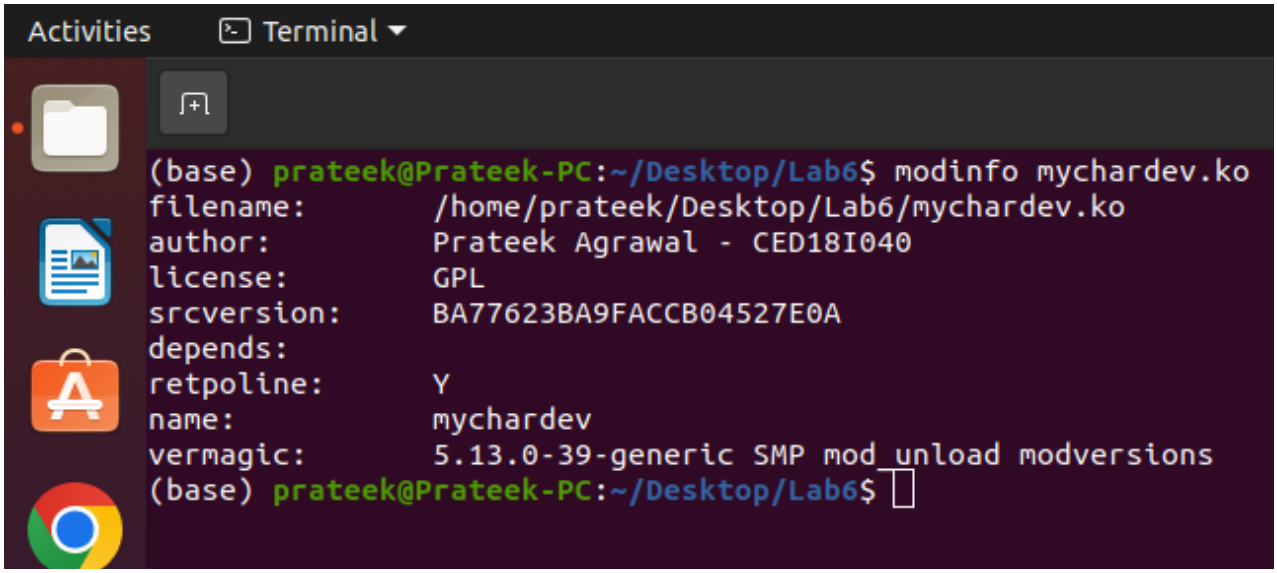
Make file :

```
BINARY     := mychardev
KERNEL      := /lib/modules/$(shell uname -r)/build
ARCH        := x86
C_FLAGS     := -Wall
KMOD_DIR    := $(shell pwd)
TARGET_PATH := /lib/modules/$(shell uname
-r)/kernel/drivers/char
```

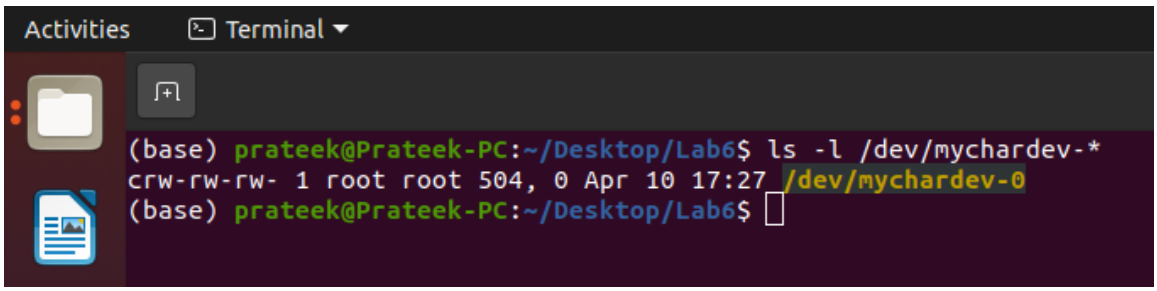
```
OBJECTS := chardd.o

ccflags-y += $(C_FLAGS)

obj-m += $(BINARY).o

$(BINARY)-y := $(OBJECTS)

$(BINARY).ko:
     make -C $(KERNEL) M=$(KMOD_DIR) modules

install:
     cp $(BINARY).ko $(TARGET_PATH)
     depmod -a

uninstall:
     rm $(TARGET_PATH)/$(BINARY).ko
     depmod -a

clean:
     make -C $(KERNEL) M=$(KMOD_DIR) clean
```

Steps :

1) Make sure that the secure boot is turned off. If not you can use the UEFI setting using mokutil to turn it off. It involves a couple of restarts but can be done.
2) Make sure that the repositories do not have blank spaces or special characters because it gives and error if they have it
3) Run the **make** command
4) Run the **sudo insmod mychardev.ko** command to insert the code into the kernel
5) Run **modinfo mychardev.ko** to get the information about the module.



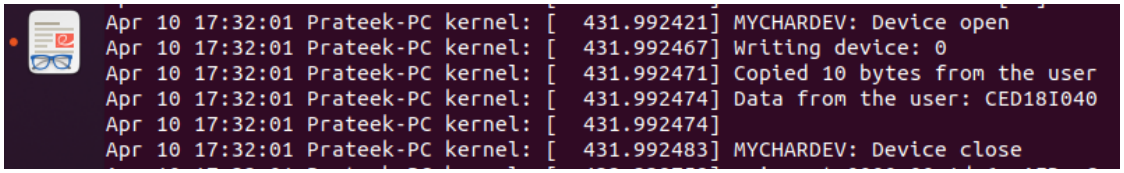
6) Check the type of device created by using **ls -l /dev/mychardev-***



7) Echo some data into the char device using echo command.
8) To check if the module has been inserted into the kernel Run

tail /var/log/kern.log

The following command prints the last few lines of the kernels log.



9) To remove the module from the kernel Run **sudo rmmod hello.ko**

Explanation:

- The first step is to Insert the module after make command using sudo
- insmod mychardev.ko
- We can check the filepath of the module using modinfo mychardev.ko
- The module has been created using the major and minor numbers **504** and **0**
- The final step is to read and write into the kernel module. Here, 10 bytes has been read and displayed on the terminal, while