Practical - 10

Aim : Implementation of a device driver to find reverse string in kernel mode.

Character Device Code :
reverse.c :

```
#include <linux/module.h>
#include <linux/init.h>
#include <linux/miscdevice.h>
#include <linux/fs.h>
#include <linux/uaccess.h>

MODULE_AUTHOR("0xe7, 0x1e");
MODULE_DESCRIPTION("A simple character device which reverses the words in a string");
MODULE_LICENSE("GPL");

#define DEVICE_SIZE 512

char data[DEVICE_SIZE+1]="no data has been written yet";

void insert_word(char *word, unsigned int n)
{
    int i, c;
    char tmpword[DEVICE_SIZE+1];
    for (i = strlen(word)-1, c = 0; i >= 0; i--, c++) {
        tmpword[c] = word[i];
    }
    tmpword[strlen(word)] = '\0';
    if (n == 0) {
        memset(data, 0, sizeof data);
        strcpy(data, tmpword);
    } else {
        data[strlen(data)] = ' ';
        data[strlen(data)+1] = '\0';
        strcat(data, tmpword);
    }
}

void reverse(char *tmpdata)
{
    int i, c;
    unsigned int n = 0;
    char word[DEVICE_SIZE+1];
    for (i = strlen(tmpdata)-1, c = 0; i >= 0; i--, c++) {
        if (tmpdata[i] == ' ') {
            word[c] = '\0';
            insert_word(word, n);
            n += 1;
            c = -1;
        } else
            word[c] = tmpdata[i];
    }
    word[c] = '\0';
    insert_word(word, n);
    data[strlen(tmpdata)] = '\0';
}

ssize_t reverse_read(struct file *filep,char *buff,size_t count,loff_t *offp )
{
    if ( copy_to_user(buff,data,strlen(data)) != 0 ) {
        printk( "Kernel -> userspace copy failed!\n" );
        return -1;
    }
    return strlen(data);
}

ssize_t reverse_write(struct file *filep,const char *buff,size_t count,loff_t *offp )
{
    char tmpdata[DEVICE_SIZE+1];
    if ( copy_from_user(tmpdata,buff,count) != 0 ) {
        printk( "Userspace -> kernel copy failed!\n" );
        return -1;
    }
    reverse(tmpdata);
    return 0;
}

struct file_operations reverse_fops = {
    read: reverse_read,
    write: reverse_write
};

static struct miscdevice reverse_misc_device = {
    .minor = MISC_DYNAMIC_MINOR,
    .name = "reverse",
    .fops = &reverse_fops
};

static int __init reverse_init(void)
{
    misc_register(&reverse_misc_device);

        return 0;
}

static void __exit reverse_exit(void)
{
    misc_deregister(&reverse_misc_device);
}

module_init(reverse_init);
module_exit(reverse_exit);
```

Makefile:

```
obj-m += reverse.o
all:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
clean:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

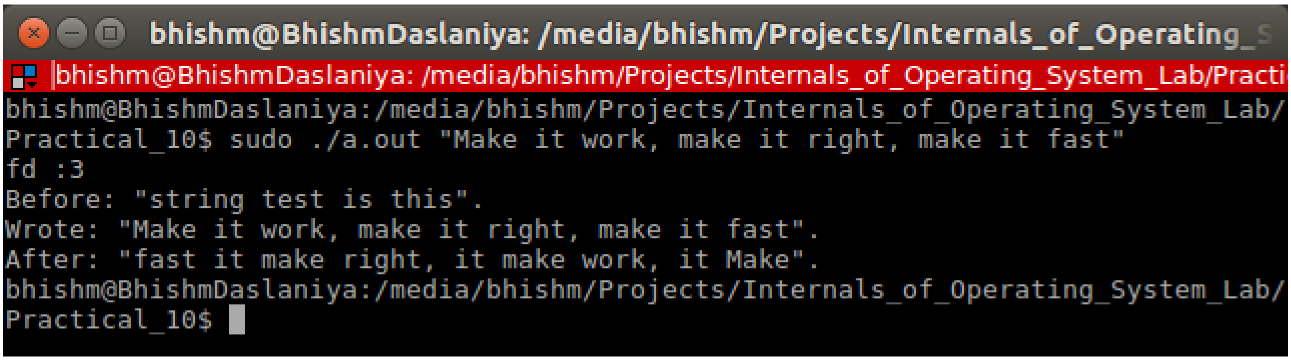
-> **run this make file using command :** make

-> **Check some modules are created. Now check reverse.ko using command :** insmod reverse.ko
If it run successfully without error then device is created and ready to use.

Application code:

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

#define CDEV_DEVICE "reverse"
static char buf[512+1];

int main(int argc, char *argv[])
{
    int fd, len;

    if (argc != 2) {
        printf("Usage: %s <string>\n", argv[0]);
        exit(0);
    }

    if ((len = strlen(argv[1]) + 1) > 512) {
        printf("ERROR: String too long\n");
        exit(0);
    }

    if ((fd = open("/dev/" CDEV_DEVICE, O_RDWR)) == -1) {
        perror("/dev/" CDEV_DEVICE);
        exit(1);
    }

    printf("fd :%d\n",fd);

    if (read(fd, buf, len) == -1)
        perror("read()");
    else
        printf("Before: \"%s\".\n", buf);

    if (write(fd, argv[1], len) == -1)
        perror("write()");
    else
        printf("Wrote: \"%s\".\n", argv[1]);

    if (read(fd, buf, len) == -1)
        perror("read()");
    else
        printf("After: \"%s\".\n", buf);

    if ((close(fd)) == -1) {
        perror("close()");
        exit(1);
    }

    exit(0);
}
```

Output:



bhishm@BhishmDaslaniya: /media/bhishm/Projects/Internals_of_Operating_S
bhishm@BhishmDaslaniya: /media/bhishm/Projects/Internals_of_Operating_System_Lab/Practi

```
bhishm@BhishmDaslaniya:/media/bhishm/Projects/Internals_of_Operating_System_Lab/
Practical_10$ sudo ./a.out "Make it work, make it right, make it fast"
fd :3
Before: "string test is this".
Wrote: "Make it work, make it right, make it fast".
After: "fast it make right, it make work, it Make".
bhishm@BhishmDaslaniya:/media/bhishm/Projects/Internals_of_Operating_System_Lab/
Practical_10$ 
```

Conclusion: From this practical I have learnt about what is kernel devices and how to make our own character device driver and use it.