Sistemas Operacionais

Desenvolvendo um Device do Tipo Caracter

André Luis Martinotto

UCS UNIVERSIDADE DE CAXIAS DO SUL

Device do Tipo Caracter – driver.c

```
/* Bibliotecas */

#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/fs.h>
#include <asm/uaccess.h>

#define DEVICE_NAME "so"
#define BUF_LEN 1024

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Andre Luis Martinotto");
MODULE_DESCRIPTION("Disciplina SO");
```

Device do Tipo Caracter – driver.c

```
/* Protótipo das funções */

/* Funcao que e executada ao inserir o modulo */
int init_module(void);
/* Funcao que e executada ao remover o modulo */
int device_init(void);
void device_exit(void);
static int device_open(struct inode *, struct file *);
static int device_release(struct inode *, struct file *);
static ssize_t device_read(struct file *, char *, size_t, loff_t *);
static ssize_t device_write(struct file *, const char *, size_t, loff_t *);

module_init(device_init);
module_exit(device_exit);
```

Device do Tipo Caracter – driver.c

```
/* Define qual funcao vai realizar qual atividade */
static struct file_operations fops = {
    .read = device_read,
    .write = device_write,
    .open = device_open,
    .release = device_release
};

/* Variáveis globais são declaradas como estáticas */
static int device = 60;
static int open = 0;
static char buffer[BUF_LEN];
static char *rptr;
static char *wptr;
```

Device do Tipo Caracter – driver.c

```
/* Funcao chamada quando o modulo e carregado */

int device_init() {
    /*cria o dispositivo*/
    int ret = register_chrdev(device, DEVICE_NAME, &fops);
    if(ret < 0) {
        printk("Nao foi possivel abrir o dispositivo %d.\n", device);
        return ret;
    }
    memset(buffer, 0, BUF_LEN);
    printk("Dispositivo carregado.\n");
    return 0;
}
```

Device do Tipo Caracter – driver.c

```
/* remove o dispositivo */
void device_exit() {
    unregister_chrdev(device, DEVICE_NAME);
    printk("Dispositivo descarregado.\n");
}
```

Device do Tipo Caracter – driver.c

```
/* Chamada quando o processo tenta abrir o dispositivo */
static int device_open(struct inode *nd, struct file *fp) {
    /*verifica se ja tem alguem acessando o dispositivo*/
    if( open ){
            return -EBUSY;
    }
    /*bloqueia o acesso ao dispositivo por outros processos*/
    open++;
    rptr = wptr = buffer;
    try_module_get(THIS_MODULE);
    return 0;
}
```

Device do Tipo Caracter – driver.c

```
/* Chamada quando um processo fecha o arquivo */
static int device_release(struct inode *nd, struct file *fp) {
    / *libera o acesso ao dispositivo */
    if(open){
            open--;
    }
    module_put(THIS_MODULE);
    return 0;
}
```

Device do Tipo Caracter – driver.c

```
/* Chamada quando um processo tenta ler o arquivo aberto */
static ssize_t device_read(struct file *fp, char *buff, size_t
                                        length, loff_t *offset) {
    int nbytes = strlen(rptr);
    if( nbytes > length){
            nbytes = length;
    }
    copy_to_user(buff, rptr, nbytes);
    rptr += nbytes;
    return nbytes;
}
```

Device do Tipo Caracter – driver.c

```
/* Chamada quando um processo tenta escrever no arquivo */
static ssize_t device_write(struct file *fp, const char *buff, size_t
                                                    length, loff_t *offset) {
    int nbytes = BUF_LEN - (wptr - buffer);
    if ( nbytes > length){
        nbytes = length;
    }
    copy_from_user(wptr, buff, nbytes);
    wptr += nbytes;
    return nbytes;
}
```

Device do Tipo Carater - Makefile

```
obj-m       :=      driver.o

KDIR        := /lib/modules/$(shell uname -r)/build
PWD         := $(shell pwd)

default:
        make -C $(KDIR) SUBDIRS=$(PWD) modules
```

Testando o Device

- **Criando o arquivo e carregando o módulo**
  - sudo mknod /dev/so c 60 0
  - sudo chmod 666 /dev/so
  - sudo insmod driver.ko
- **Testando o módulo**
  - echo "Alo mundo" > /dev/so
  - cat /dev/so
- **Removendo o módulo e o arquivo**
  - sudo rmmod driver.ko
  - sudo rm /dev/so

Acessando o Device

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>

#define DEVICE_NAME "/dev/so"
#define BUF_LEN 1024

int main(){
        int file;
        char wrt[BUF_LEN]  = "Acessando o driver!!!\n";
        char rd[BUF_LEN];

        file = open(DEVICE_NAME, O_RDWR);
```

Acessando o Device

```
        if (file < 0) {
                printf("Erro ao abrir o device!!!\n");
                exit(-1);
        }

        write(file, wrt, BUF_LEN);
        read(file, rd, BUF_LEN);
        printf("%s\n",rd);
        close(file);
}
```