


# Sistemas Operativos

# Práctica 4

Curso

2018-2019

## Uso de un driver para una video-consola en Linux

Carlos García

# Índice

# Introducción

- Objetivos:
  - Comprender el funcionamiento de un driver
  - Creación de un driver, montado y desmontado del módulo del kernel
  - Interacción de un usuario con un driver para un mando de vídeo-consola

# Introducción

- Recordad:
  - Compilación del driver
  - Montado y desmontado de driver mediante un módulo del kernel

# Introducción

| **Aplicaciones** | **Módulos** |
|---|---|
| Modo usuario | Modo kernel |
| Cualquier función de biblioteca disponible | Sólo símbolos exportados por el kernel:<br>• /proc/kallsyms<br>• libc no disponible<br>• printk() |
| Realizan su función de principio a fin (main) | Función de inicio: "init_module"<br>Función de fin: "cleanup_module" |

- En los módulos no podemos usar printf (libc).
- El kernel proporciona una función similar: printk

# Índice

# Ejercicio 1

hello.c

```c
/*
 *  hello.c - The simplest kernel module.
 */
#include <linux/module.h> /* Needed by all modules */
#include <linux/kernel.h> /* Needed for KERN_INFO */

int init_module(void)
{
    printk(KERN_INFO "Hello world.\n");

    /*
     * A non 0 return means init_module failed; module can't be
  loaded.
     */
    return 0;
}

void cleanup_module(void)
{
    printk(KERN_INFO "Goodbye world.\n");
}
```

# Ejercicio 1

- Los módulos del kernel deben compilarse de forma diferente a los ficheros C habituales.

**makefile**

```
obj-m += hello.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
```

# Ejercicio 1

**Terminal #1**

```
  usuarioso@ssoo:~$ make
make -C /lib/modules/4.15.0-24-generic/build M=/tmp/FicherosP4/Hello modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-24-generic'
  CC [M]  /tmp/FicherosP4/Hello/hello.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /tmp/FicherosP4/Hello/hello.mod.o
  LD [M]  /tmp/FicherosP4/Hello/hello.ko
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-24-generic'

  usuarioso@ssoo:~$ sudo insmod hello.ko
  [sudo] password for usuarioso:
  usuarioso@ssoo:~$  lsmod | grep hello
hello                  16384  0

  usuarioso@ssoo:~$ sudo rmmod hello
```

# Ejercicio 2: chardev

- Implementar un driver como módulo
  - Crear un módulo del kernel con funciones `init_module()` y `cleanup_module()`
  - Implementar las operaciones de la interfaz del dispositivo de caracteres: `struct file_operations`
  - En la función de inicialización
    - Crear una estructura `cdev_t` y asociarle las operaciones y el rango de major/minor
    - Reservar major number y rango de minor numbers para el driver `alloc_chrdev_region()`
    - Usar `cdev_alloc()`, `cdev_init()` y `cdev_add()`
  - En la función de descarga
    - Destruir estructura cdev_t: `cdev_del()`
    - Liberar el rango (major, minor): `unregister_chrdev_region()`

# Ejercicio 2

## struct file_operations

```
struct file_operations {
  struct module *owner;
  loff_t(*llseek) (struct file *, loff_t, int);
  ssize_t(*read) (struct file *, char __user *, size_t,
    loff_t *);
  ssize_t(*write) (struct file *, const char __user *, size_t,
    loff_t *);
  int (*readdir) (struct file *, void *, filldir_t);
  int (*ioctl) (struct inode *, struct file *, unsigned int,
    unsigned long);
  int (*mmap) (struct file *, struct vm_area_struct *);
  int (*open) (struct inode *, struct file *);
  int (*flush) (struct file *);
  int (*release) (struct inode *, struct file *);
  int (*fsync) (struct file *, struct dentry *, int datasync);
  int (*lock) (struct file *, int, struct file_lock *);
  ssize_t(*readv) (struct file *, const struct iovec *,
    unsigned long, loff_t *);
  ssize_t(*writev) (struct file *, const struct iovec *,
    unsigned long, loff_t *);
  ssize_t(*sendfile) (struct file *, loff_t *, size_t,
    read_actor_t, void __user *);
  ssize_t(*sendpage) (struct file *, struct page *, int, size_t,
    loff_t *, int);
  // ...
};
```

# Ejercicio 2

- No todos los campos de esta estructura deben inicializarse
- Sólamente aquellas que se corresponden con operaciones soportadas por el driver

**struct file_operations**

```
struct file_operations fops = {
  .read = device_read,
  .write = device_write,
  .open = device_open,
  .release = device_release
};
```

# Ejercicio 2

chardev.c

```
/*
 *  Prototypes
 */
int init_module(void);
void cleanup_module(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 *);

struct file_operations fops = {
  .read = device_read,
  .write = device_write,
  .open = device_open,
  .release = device_release
};
```

# Ejercicio 2

**init_module en chardev.c**

```
int init_module(void)
{
    int major;    /* Major number assigned to our device driver */
    int minor;    /* Minor number assigned to the character device
  */
    int ret;

    /* Get available (major,minor) range */
    if ((ret=alloc_chrdev_region (&start, 0, 1,DEVICE_NAME))) {
        printk(KERN_INFO "Can't allocate chrdev_region()");
        return ret;
    }

    /* Create associated cdev */
    if ((chardev=cdev_alloc())==NULL) {
        printk(KERN_INFO "cdev_alloc() failed ");
        unregister_chrdev_region(start, 1);
        return -ENOMEM;
    }

    cdev_init(chardev,&fops);

    if ((ret=cdev_add(chardev,start,1))) {
        printk(KERN_INFO "cdev_add() failed ");
        kobject_put(&chardev->kobj);
        unregister_chrdev_region(start, 1);
        return ret;
    }

    .....
```

# Ejercicio 2

## init_module en chardev.c

```
int init_module(void)
{
    ....

    major=MAJOR(start);
    minor=MINOR(start);

    printk(KERN_INFO "I was assigned major number %d. To talk to\n"
  , major);
    printk(KERN_INFO "the driver, create a dev file with\n");
    printk(KERN_INFO "'sudo mknod -m 666 /dev/%s c %d %d'.\n",
  DEVICE_NAME, major,minor);
    printk(KERN_INFO "Try to cat and echo to the device file.\n");
    printk(KERN_INFO "Remove the device file and module when done.\
  n");

    return SUCCESS;
}
```

# Ejercicio 2

**Terminal #1**

```
usuarioso@ssoo:~$ sudo insmod chardev.ko
usuarioso@ssoo:~$ dmesg | tail -n 5
[ 6227.827687] I was assigned major number 243. To talk to
[ 6227.827694] the driver, create a dev file with
[ 6227.827698] 'sudo mknod -m 666 /dev/chardev c 243 0'.
[ 6227.827700] Try to cat and echo to the device file.
[ 6227.827702] Remove the device file and module when done.
```

# Ejercicio 2

- Incrementa/Decrementa contador interno de uso (si contador !=0 no puede eliminarse)
  - try_module_get(THIS_MODULE);
  - module_put(THIS_MODULE);

**device_open y device_release en chardev.c**

```
/*
 * Called when a process tries to open the device file, like
 * "cat /dev/chardev"
 */
static int device_open(struct inode *inode, struct file *file)
{
    if (Device_Open)
        return -EBUSY;

    Device_Open++;

    /* Initialize msg */
    sprintf(msg, "I already told you %d times Hello world!\n",
  counter++);

    /* Initially, this points to the beginning of the message */
    msg_Ptr = msg;

    /* Increase the module's reference counter */
    try_module_get(THIS_MODULE);

    return SUCCESS;
}
```

# Ejercicio 2

- Incrementa/Decrementa contador interno de uso (si contador !=0 no puede eliminarse)
  - try_module_get(THIS_MODULE);
  - module_put(THIS_MODULE);

**device_open y device_release en chardev.c**

```
/*
 * Called when a process closes the device file.
 */
static int device_release(struct inode *inode, struct file *file)
{
    Device_Open--;    /* We're now ready for our next caller */

    /*
     * Decrement the usage count, or else once you opened the file,
   you'll
     * never get get rid of the module.
     */
    module_put(THIS_MODULE);

    return 0;
}
```

# Ejercicio 2

- Las operaciones read y write de un fichero de dispositivo tienen como parámetro un puntero buffer del espacio de usuario
  - Siempre hay que trabajar con copia privada en el espacio del kernel
    - copy_from_user() y copy_to_user()

copy_to_user en chardev.c

```
/*
 * Called when a process, which already opened the dev file,
  attempts to
 * read from it.
 */
static ssize_t device_read(struct file *filp,/* see include/linux/
  fs.h   */
                    char *buffer, /* buffer to fill with data */
                    size_t length,  /* length of the buffer     */
                    loff_t * offset)
{
  ....

    /*
     * Actually transfer the data onto the userspace buffer.
     * For this task we use copy_to_user() due to security issues
     */
    if (copy_to_user(buffer,msg_Ptr,bytes_to_read))
        return -EFAULT;

  ....

    return bytes_to_read;
}
```

# Ejercicio 2

copy_to_user en chardev.c

```
/*
 * Called when a process, which already opened the dev file,
  attempts to
 * read from it.
 */
static ssize_t device_read(struct file *filp,
                    char *buffer,
                    size_t length,
                    loff_t * offset)
{
    /*
     * Number of bytes actually written to the buffer
     */
    int bytes_to_read = length;

    /*
     * If we're at the end of the message,
     * return 0 -> end of file
     */
    if (*msg_Ptr == 0)
        return 0;

    /* Make sure we don't read more chars than
     * those remaining to read
       */
    if (bytes_to_read > strlen(msg_Ptr))
        bytes_to_read=strlen(msg_Ptr);

    /*
     * Actually transfer the data onto the userspace buffer.
     * For this task we use copy_to_user() due to security issues
     */
    if (copy_to_user(buffer,msg_Ptr,bytes_to_read))
        return -EFAULT;

    /* Update the pointer for the next read operation */
    msg_Ptr+=bytes_to_read;

    /*
     * The read operation returns the actual number of bytes
     * we copied  in the user's buffer
     */
    return bytes_to_read;
}
```

# Ejercicio 2

## Terminal #1

```
  usuarioso@ssoo:~$ sudo insmod chardev.ko
  usuarioso@ssoo:~$ dmesg | tail -n 5
[ 6227.827687] I was assigned major number 243. To talk to
[ 6227.827694] the driver, create a dev file with
[ 6227.827698] 'sudo mknod -m 666 /dev/chardev c 243 0'.
[ 6227.827700] Try to cat and echo to the device file.
[ 6227.827702] Remove the device file and module when done.

  usuarioso@ssoo:~$ sudo mknod -m 666 /dev/chardev c 243 0
  usuarioso@ssoo:~$ cat /dev/chardev
I already told you 1 times Hello world!

  usuarioso@ssoo:~$ cat /dev/chardev
I already told you 2 times Hello world!

  usuarioso@ssoo:~$ echo 1 > /dev/chardev
bash: echo: write error: Operation not permitted

  usuarioso@ssoo:~$ sudo rmmod chardev
```

# Ejercicio 3: Xbox 360

- Uso del driver *xpad* correspondiente al mando de la vídeo-consola XBox-360
- Se carga por defecto al conectarse el mando
- El controlador crea tres dispositivos:
  - `/dev/input/jsX`: evento producidos por pulsación de palanca o botón
  - `/sys/class/leds/xpadX/brightness`: activación/desactivación de los leds
  - `/dev/input/eventX`: modo de vibración del mando

# Ejercicio 3: Xbox 360

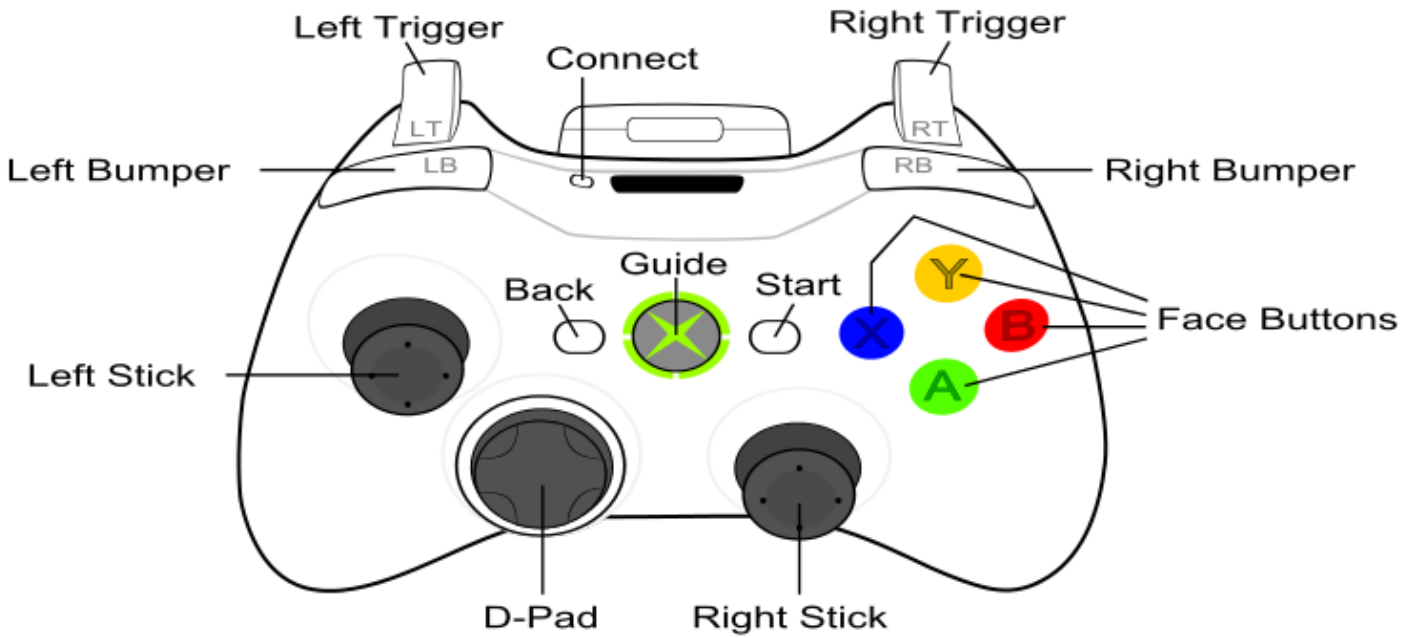


**Figura:** Detalle del mando X-Box 360

# Ejercicio 3: Xbox 360

- Evento producidos por pulsación de palanca o botón
  - Se puede saber mediante la aplicación *jstest*

**Terminal #1**

```
usuarioso@ssoo:~$ ./jstest /dev/input/js0
Joystick (Microsoft X-Box 360 pad) has 8 axes and 11 buttons. Driver version is 2.1.0.
Testing ... (interrupt to exit)
Axes:  0:  1832  1: -3367  2:-32767  3:  5590  4:    91  5:-32767  6:     0  7:     0
Buttons:  0:off  1:off  2:off  3:off  4:off  5:off  6:off  7:off  8:off  9:off 10:off
```

**Cuadro:** Correspondencia entre el identificador y el botón correspondiente al mando X-Box 360

| id | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| botón | A | B | X | Y | left bumper | right bumper | back | start | guide |

# Ejercicio 3: Xbox 360

- Activación de leds del botón *Guide*
  - Se lleva a cabo mediante la escritura en el dispositivo `/sys/class/leds/xpadX/brightness`
  - El valor escrito en el dispositivo diferentes configuraciones leds
    - 0: apagado
    - 1: todos leds parpadean, después configuración previa
    - 2: parpadeo el led superior izquierdo, luego se queda encendido
    - 3: parpadeo el led superior derecho, luego se queda encendido
    - 4: parpadeo el led inferior izquierdo, luego se queda encendido
    - 5: parpadeo el led inferior derecho, luego se queda encendido
    - 6: led superior izquierdo encendido
    - 7: led superior derecho encendido
    - 8: led inferior izquierdo encendido
    - 9: led superior derecho encendido
    - 10: leds en rotación
    - 11: led parapadean según la configuración anterior
    - 12: led parapadean lentamente según la configuración anterior
    - 13: dos leds en rotación, después configuración previa
    - 14: todos los leds parpadean
    - 15: todos los leds parpadean, después configuración previa

Terminal #1

```
usuarioso@ssoo:~$ echo 9 > /sys/class/leds/xpad0/brightness
```

# Ejercicio 3: Xbox 360

- Vibración mediante escritura en `/dev/input/eventX`
  - Uso de aplicación *fftest*
    - Bucle de vibración sinusoidad de intensidad creciente
      - Intensidad: 0 %-100 % con parámetro `gain.value` (0, FFFF)
      - Paramétros de vibración sinusoidad en estructura `effect`
    - Vibración tipo `rumble` de ambos motores del mando
      - Vibración del mando izquierdo (`strong`) y derecho (`weak`)

Terminal #1

```
usuarioso@ssoo:~$ ./fftest /dev/input/event23
```

# Índice

# Leds del mando y su vibración

- Creación de aplicación **xbox_test**:
  - Se inicializará el mando con led superior izquierdo encendido y sin niguna vibración.
  - Si se pulsa el botón "start", el mando volverá a su configuración inicial.
  - Si se pulsa el botón "A" el activará la vibración de la izquierda (vibración `strong`)
  - Si se pulsa el botón "Y" el activará la vibración de la derecha (vibración `weak`)
  - Si se pulsa el botón "B" el led girará hacia la derecha.
  - Si se pulsa el botón "X" el led girará hacia la izquierda.
  - Si se pulsa el botón "left bumper" se decrementará la vibración.
  - Si se pulsa el botón "right bumper" se incrementará la vibración.

# Disp. asociados con Xbox 360

- Consultando el fichero `/proc/bus/input/devices`
  - En apartado Handlers:
    - Dispositivo "evento" **event23**: eventos en `/dev/input/event23`
    - Dispositivo de entrada **js0**: `/dev/input/js0`

**Terminal #1**

```
usuarioso@ssoo:~$ cat /proc/bus/input/devices

I: Bus=0003 Vendor=045e Product=028e Version=0114
N: Name="Microsoft X-Box 360 pad"
P: Phys=usb-0000:06:00.0-1/input0
S: Sysfs=/devices/pci0000:00/0000:00:1c.6/0000:06:00.0/usb5/5-1/5-1:1.0/input/input27
U: Uniq=
H: Handlers=event23 js0
B: PROP=0
B: EV=20000b
B: KEY=7cdb000000000000 0 0 0 0
B: ABS=3003f
B: FF=107030000 0
```