



UNIVERSIDAD AUTONOMA DE CHIHUAHUA  
FACULTAD DE INGENIERIA

SISTEMAS OPERATIVOS I  
*IVAN MIGUEL CHAVERO JURADO*

EXAMEN FINAL – TERCER PARCIAL

ALUMNA

345498

ARIDAHÍ GÁMEZ ESCUDERO

31 de Mayo del 2022

## ANTES QUE NADA

Tuve un error al compilar el código el cual no me permitió realizar las instrucciones, sin embargo, las hice robando capturas del código de mi compañero Jorge en clase, lamento el inconveniente.

```
[defectum@fedora Descargas]$ make
echo "Compilando modulo Examen Parcial Kernel"
Compilando modulo Examen Parcial Ker
make -C /lib/modules/5.17.11-200.fc35.x86_64/Descargas modules
make[1]: se entra en el directorio '/usr/src/kernels/5.17.11-200.fc35.x86_64'
warning: the compiler differs from the one used to build the kernel
The kernel was built by: gcc (GCC) 11.3.1 20220421 (Red Hat 11.3.1-2)
You are using:          gcc (GCC) 11.2.1 20210728 (Red Hat 11.2.1-1)
CC [M] /home/defectum/Descargas/kernel_module.o
gcc: error: unrecognized command-line option '-mharden-ssls=all'
make[2]: *** [scripts/Makefile.build:288: /home/defectum/Descargas/kernel_module.o] Error 1
make[1]: *** [Makefile:1841: /home/defectum/Descargas] Error 2
make[1]: se sale del directorio '/usr/src/kernels/5.17.11-200.fc35.x86_64'
make: *** [Makefile:5: all] Error 2
```

## INTRODUCCIÓN

En esta ocasión haremos uso de un código que nos permita agregar un módulo al kernel y haremos uso de este para darnos a la tarea de repetir la frase de una canción en el espacio de usuario hasta que hagamos la interrupción del programa.

## CODIGO

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

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Aridahi Gamez");
MODULE_DESCRIPTION("Examen Parcial Kernel");
MODULE_VERSION("0.01");

#define DEVICE_NAME "kernel_module"

static int major;

static int dev_open(struct inode*, struct file*);
static int dev_release(struct inode*, struct file*);
static ssize_t dev_read(struct file*, char*, size_t, loff_t*);
static ssize_t dev_write(struct file*, const char*, size_t, loff_t*);
```

```

static struct file_operations fops = {
    .open = dev_open,
    .read = dev_read,
    .write = dev_write,
    .release = dev_release,
};

static int __init kernel_module_init(void) {
    major = register_chrdev(0, DEVICE_NAME, &fops);

    if (major < 0) {
        printk(KERN_ALERT "Examen Parcial Kernel load failed\n");
        return major;
    }

    printk(KERN_INFO "Inicializando módulo Examen Parcial Kernel!!!\n");
    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 "'mknod /dev/%s c %d 0'.\n", DEVICE_NAME, major);
    printk(KERN_INFO "Try various minor numbers. Try to cat and echo
to\n");
    printk(KERN_INFO "the device file.\n");
    printk(KERN_INFO "Remove the device file and module when done.\n");

    return 0;
}

static int dev_open(struct inode *inodep, struct file *filep) {
    printk(KERN_INFO "Examen 3er Parcial device opened\n");
    return 0;
}

static ssize_t dev_write(struct file *filep, const char *buffer,
                        size_t len, loff_t *offset) {

    printk(KERN_INFO "Sorry, Examen Parcial Kernel is read only\n");
    return -EFAULT;
}

static int dev_release(struct inode *inodep, struct file *filep) {
    printk(KERN_INFO "Examen Parcial Kernel device closed\n");
    return 0;
}

```

```
static ssize_t dev_read(struct file *filep, char *buffer, size_t len,
loff_t *offset) {
    int errors = 0;
    char *message = "Break you in a thousand parts, used to be a shooting
star... ";
    int message_len = strlen(message);

    errors = copy_to_user(buffer, message, message_len);

    return errors == 0 ? message_len : -EFAULT;
    return 0;
}

static void __exit kernel_module_exit(void) {
    printk(KERN_INFO "Desacoplando módulo Examen Parcial Kernel!\n");
}

module_init(kernel_module_init);
module_exit(kernel_module_exit);
```

## PROCEDIMIENTO

En esta ocasión utilizaremos un archivo Makefile para poder compilar nuestro código.

```
obj-m += kernel_module.o

all:
    echo "Compilando modulo Examen Parcial Kernel"
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

Debemos asegurarnos de que el nombre de nuestro código se encuentra capturado en nuestro Makefile, una vez listo nos iremos a la terminal donde usaremos la instrucción:

**Make** *instrucción utilizada para compilar*

```
[jorge@fedora Universidad]$ make
echo "Compilando modulo batman"
Compilando modulo batman
make -C /lib/modules/5.17.9-200.fc35.x86_64/build M=/home/jorge/Universidad modules
make[1]: Entering directory '/usr/src/kernels/5.17.9-200.fc35.x86_64'
CC [M] /home/jorge/Universidad/Batman.o
MODPOST /home/jorge/Universidad/Module.symvers
CC [M] /home/jorge/Universidad/Batman.mod.o
LD [M] /home/jorge/Universidad/Batman.ko
BTF [M] /home/jorge/Universidad/Batman.ko
Skipping BTF generation for /home/jorge/Universidad/Batman.ko due to unavailability of vmlinux
```

*Imagen ilustrativa*

Una vez estos datos estén en pantalla podremos saber que nuestro código ha compilado con éxito. Posteriormente utilizaremos la siguiente instrucción para insertar nuestro modulo en el kernel, debemos recordar el agregar la palabra sudo para poder ejecutar la instrucción con el privilegio de administrador que nos ofrece el comando:

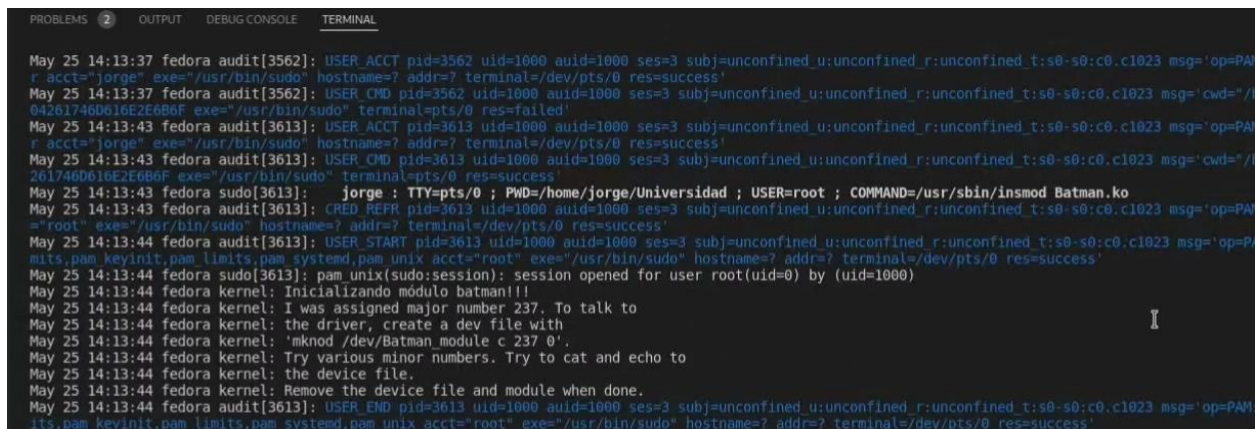
## sudo ismod kernel.ko

*sudo: privilegio de administrador*

*ismod: inserta el módulo en el kernel*

*kernel.ko: nombre de nuestro archivo con código*

Una vez hecho esto pasaremos a utilizar una segunda terminal donde colocaremos la instrucción **journalctl -f** la cual nos permite acceder a los registros del sistema filtrando los mensajes en tiempo real.



```
May 25 14:13:37 fedora audit[3562]: USER ACCT pid=3562 uid=1000 auid=1000 ses=3 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 msg='op=PAI
r acct="jorge" exe="/usr/bin/sudo" hostname=? addr=? terminal=/dev/pts/0 res=success'
May 25 14:13:37 fedora audit[3562]: USER_CMD pid=3562 uid=1000 auid=1000 ses=3 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 msg='cwd="/i
042617460616E2E686F exe="/usr/bin/sudo" terminal=pts/0 res=failed'
May 25 14:13:43 fedora audit[3613]: USER ACCT pid=3613 uid=1000 auid=1000 ses=3 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 msg='op=PAI
r acct="jorge" exe="/usr/bin/sudo" hostname=? addr=? terminal=/dev/pts/0 res=success'
May 25 14:13:43 fedora audit[3613]: USER_CMD pid=3613 uid=1000 auid=1000 ses=3 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 msg='cwd="/i
2617460616E2E686F exe="/usr/bin/sudo" terminal=pts/0 res=success'
May 25 14:13:43 fedora sudo[3613]: jorge : TTY=pts/0 ; PWD=/home/jorge/Universidad ; USER=root ; COMMAND=/usr/sbin/insmod Batman.ko
May 25 14:13:43 fedora audit[3613]: CRED REPR pid=3613 uid=1000 auid=1000 ses=3 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 msg='op=PAI
="root" exe="/usr/bin/sudo" hostname=? addr=? terminal=/dev/pts/0 res=success'
May 25 14:13:44 fedora audit[3613]: USER_START pid=3613 uid=1000 auid=1000 ses=3 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 msg='op=P
mits.pam keyinit.pam limits.pam systemd.pam unix acct="root" exe="/usr/bin/sudo" hostname=? addr=? terminal=/dev/pts/0 res=success'
May 25 14:13:44 fedora sudo[3613]: pam_unix(sudo:session): session opened for user root(uid=0) by (uid=1000)
May 25 14:13:44 fedora kernel: Inicializando módulo batman!!!
May 25 14:13:44 fedora kernel: I was assigned major number 237. To talk to
May 25 14:13:44 fedora kernel: the driver, create a dev file with
May 25 14:13:44 fedora kernel: 'mknod /dev/Batman.module c 237 0'.
May 25 14:13:44 fedora kernel: Try various minor numbers. Try to cat and echo to
May 25 14:13:44 fedora kernel: the device file.
May 25 14:13:44 fedora kernel: Remove the device file and module when done.
May 25 14:13:44 fedora audit[3613]: USER_END pid=3613 uid=1000 auid=1000 ses=3 subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 msg='op=PAI
its.pam keyinit.pam limits.pam systemd.pam unix acct="root" exe="/usr/bin/sudo" hostname=? addr=? terminal=/dev/pts/0 res=success'
```

*Imagen ilustrativa*

Una vez aquí localizaremos la línea que contenga “mknod”; el cual crea ficheros especiales de bloques o caracteres, para copiarle y regresar a nuestra primera terminal donde pegaremos la línea en compañía de un sudo.

sudo mknod /dev/kernel\_module c 237 0

Y finalmente tendremos nuestro modulo, para poder ejecutar el texto indicado usaremos la instrucción:

cat /dev/kernel\_module

[illegible]

*imagen ilustrativa*

Para detener la reproducción del texto haremos uso de las teclas Control + C.