Universidad Autónoma de Chihuahua

Sistemas Operativos I

Proyecto Final

- Jadir Alexis Levario Ojeda – 362243
- Alejandro Morales Rodríguez – 357796
- Ariana Janeth Franco Ríos – 357756
- Andree Javier Ordaz Chavira - 360844

Chihuahua, Chih 05/12/2023

¿Cómo funciona nuestro código?

Pues….

Este código es un módulo de kernel de Linux que proporciona un dispositivo de caracteres.

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

Incluye las cabeceras necesarias para el desarrollo de módulos de kernel de Linux.

```
MODULE_LICENSE("GPL");
MODULE_AUTHOR("Team4evah");
MODULE_DESCRIPTION("Modulo anti-Microsoft!");
MODULE_VERSION("0.01");
```

Nos proporciona información sobre el módulo, como la licencia, el autor, la descripción y la versión.

```
#define DEVICE_NAME "Team4evah"
```

Define el nombre del dispositivo que se utilizará en el módulo.

```
static int major;
struct cdev char_device;
```

```
char user_message[1024] = "(Reggaetón, champán, -pán-pán-pán, -pán, -pán)
```

- major: Almacena el número principal asignado al dispositivo.
- char_device: Estructura cdev utilizada para representar el dispositivo de caracteres.
- user_message: Buffer para almacenar mensajes del usuario.

```
static int Team4evah_release(struct inode*, struct file*);
static ssize_t Team4evah_read(struct file*, char*, size_t, loff_t*);
static ssize_t Team4evah_write(struct file*, const char*, size_t, loff_t*);
int write_to_user(char *, char *);
```

Declaraciones de funciones que serán utilizadas en el módulo.

```
static struct file_operations fops = {
    .owner = THIS_MODULE,
    .release = Team4evah_release,
    .read = Team4evah_read,
    .write = Team4evah_write,
};
```

Estructura que define las operaciones que se pueden realizar en el archivo, como leer, escribir y liberar.

Se implementan las funciones:

- Team4evah_write: Lee datos del espacio de usuario y los almacena en user_message.
- Team4evah_release: Se llama cuando el archivo se cierra.
- Team4evah_read: Devuelve un mensaje predefinido al usuario.

```
static int __init Team4evah_init(void) {
```

```
static void __exit Team4evah_exit(void) {
```

Funciones llamadas al cargar y descargar el módulo, respectivamente.

```
static int __init Team4evah_init(void) {
    int result;
    major = register_chrdev(0, DEVICE_NAME, &fops);

    if (major < 0) {
        printk(KERN_ALERT "Team4evah load failed\n");
        return major;
    }

    printk(KERN_INFO "Team4evah Module has been loaded\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 void __exit Team4evah_exit(void) {
    printk(KERN_INFO "Siempre seremos un team, removiendo módulo!!\n");
}
```

En la función de inicialización, se registra el módulo y se imprime información sobre cómo interactuar con el dispositivo.

```
module_init(Team4evah_init);
module_exit(Team4evah_exit);
```

Macros que especifican las funciones de inicio y salida del módulo.

Este módulo funciona de manera similar al anterior, proporcionando un dispositivo de caracteres que permite a los usuarios leer y escribir mensajes. El contenido de user_message es un conjunto extenso de letras de una canción.