# Drivers

## Sistemas Operativos II

Alumna : Saczkowski, Sabrina Evelyn.

Índice

# Enunciado Práctico

Elaborar un kernel module para un char device, un tipo de dispositivo caracterizado por que pueden accederse mediante una tira de bytes y que se los usa directamente mediante un nodo en el filesystem.

Se debe definir las siguientes funciones: **init_module**, **cleanup_module ,** **device_open** y **device_release.**

Al escribirle al char device debe imprimir en el kernel lo último que fue escrito. Utilizando el material de lectura *The Linux Kernel Module Programming Guide* que se puede acceder mediante el siguiente enlace.

# Código Fuente

El código fuente se encuentra alojado en el siguiente repositorio:

Github: https://github.com/ssaczkowski/sor2-ssaczkowski.git

# Implementación

Para crear un char device con la funcionalidad estilo CTRL-C / CTRL-V se definió un device de la siguiente manera:

Se creó la función de inicialización **init_module** que se invoca apenas se inserta el módulo al kernel y realiza el registro del char device obteniendo dinámicamente un valor para el número **major** (valor asociado a un driver en particular), de esta forma se evitan problemas que aparecen al crearse manualmente y como **minor** se usa “0”.

Luego, se definió las funciones del device: **device_read**, **device_write**, **device_open**, **device_release**.

Se utiliza una variable global estática para manejar el traspaso de la información del mensaje con un máximo de buffer para no lidiar con administración de memoria.

Además, se invocan las funciones **get_user()** y **put_user(),** para obtener y devolver la información del espacio de usuario. Esto es debido a que se está ejecutando a nivel de kernel con máximo privilegio y es el mecanismo para interactuar con el espacio del usuario.

Se definió la función **device_write** que se invoca cuando se lanza la instrucción, “echo "Hola Mundo" > /dev/UNGS” (por ejemplo) que obtiene la secuencia de char del

espacio de usuario y la guarda en una variable colocando como final de la secuencia el valor **'\0'** para que de esa forma no guardar datos por demás.

Se desarrolla la función **device_open** que se ejecuta cuando se lanza la instrucción “cat /dev/UNGS”. Esta función invoca a **device_read** que se encarga de realizar la lectura del buffer y dejarla accesible para el espacio de usuario y así poder mostrarla por consola mediante la función **sprintf**. Se utilizó una variable **Device_Open** para controlar la apertura del device.

Si bien existe el mecanismo de crear el nodo de filesystem mediante funciones desde el módulo, se optó por utilizar el comando **dmesg** (log del kernel) para visualizar los pasos a seguir para crear el nodo y otorgarle los permisos necesarios manualmente.

# Pasos de Ejecución y Requisitos

Para que la ejecución de los comandos sea exitosa se requiere tener instalado en la VM:

1. make
2. module-init-tools
3. linux-headers- ( sale de uname -r )

Luego, para ejecutar y probar el char device se debe:

1. Descargar código del repositorio.
2. Dentro de la carpeta **sor2-ssaczkowski** , abrir terminal y hacer **>> make clean** y luego **>> make**.
3. En modo **SU**.
4. Instalar el módulo **>> sudo insmod ungsdev.ko.** Se puede verificar su correcta carga mediante **>> lsmod | grep ungsdev.** En caso de tener el módulo cargado, descargarlo con **>> rmmod ungsdev**.
5. Acceder al log del kernel para visualizar los próximos pasos. **>> dmesg**
6. Continuar los pasos del log obtenidos en el punto (4) para crear un nodo en el filesystem y usar el device mediante los comandos **echo** y **cat**.

# Recursos

http://albertomorales.eu/blog/creacion-y-puesta-a-punto-de-una-maquina-virtual-virtualbox-con-lubuntu-en-6-pasos-tiempo-inferior-a-10-min/

https://www.osboxes.org/virtualbox-images/

https://stackoverflow.com/questions/1735919/whats-the-best-way-to-reset-a-char-in-c