Universidad Nacional de General Sarmiento

Sistemas Operativos y Redes 2

Trabajo práctico Char Device

Profesores:

- Agustín Alexander
- Pedro Gutierrez

Alumna:

Dougan, Vanesa Solange

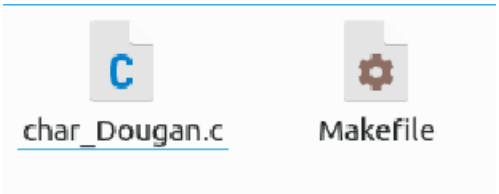
Contenido

Objetivo

Crear un módulo de Kernel para un dispositivo de carácter. Los dispositivos de caracteres manejan flujos de caracteres, por lo tanto cuando reciben la información la usan y la transmiten sin almacenarla. Por ejemplo, una placa de red recibe datos y va distribuyéndolos. No cuentan con la función SEEK.
Se deberá poder escribir en el dispositivo de carácter creado y al leer lo escrito deberá devolver el mensaje cifrado con Cesar.

Estructura del proyecto

El proyecto tiene dos archivos, un archivo con extensión .c y un MakeFile.



MakeFile

Dentro de este archivo se encuentran las configuraciones para poder compilar el módulo char_Dougan.o.

Char_dougan.c

Dentro de este archivo se encuentran las funciones necesarias que necesita el módulo Kernel .

int init_module(void)

Esta función es esencial para que funcione. Se encarga de comunicarle al Kernel qué funcionalidad provee el módulo y realizar la configuración. Dentro de esta función, por medio de register_chrdev se registra el dispositivo pasándole un 0 como major (que significa que debe asignarle dinámicamente el *major number*), el nombre del dispositivo "DOUGAN" y un puntero a al file *operation table*. El *minor number* no lo tiene en cuenta.
Esta función es llamada cuando se ejecuta insmod (instalamos el modulo).

void cleanup_module(void)

Esta función también es esencial. Se encarga de des-registrar la funcionalidad que la función init registró. Por medio de unregister_chrdev utilizando el *major number* y el nombre del dispositivo "DOUGAN" realiza la des-registración.
Es llamada cuando se ejecuta rmmod (remover el módulo).

static int device_open(struct inode *, struct file *)

Permite realizar la apertura del archivo del dispositivo.

static int device_release(struct inode *, struct file *)

Permite realizar el cierre del archivo del dispositivo.

static ssize_t device_read(struct file *, char *, size_t, loff_t *);

Se encarga de leer del dispositivo.
Por medio de put_user (valorACopiarAUserSpace, Fuente) copia los datos del segmento de datos del kernel al segmento de datos del usuario.
Es llamada cuando se ejecuta cat.

static ssize_t device_write(struct file *, const char *, size_t, loff_t *)

Se encarga de escribir en el dispositivo.
Por medio de get_user (VariableQueAlmacenaResultado, FuenteUserSpace) se obtiene la variable del modo usuario al modo Kernel.
Dentro de esta función se agregó el cifrado utilizando Cesar.
Es llamada cuando se ejecuta echo.

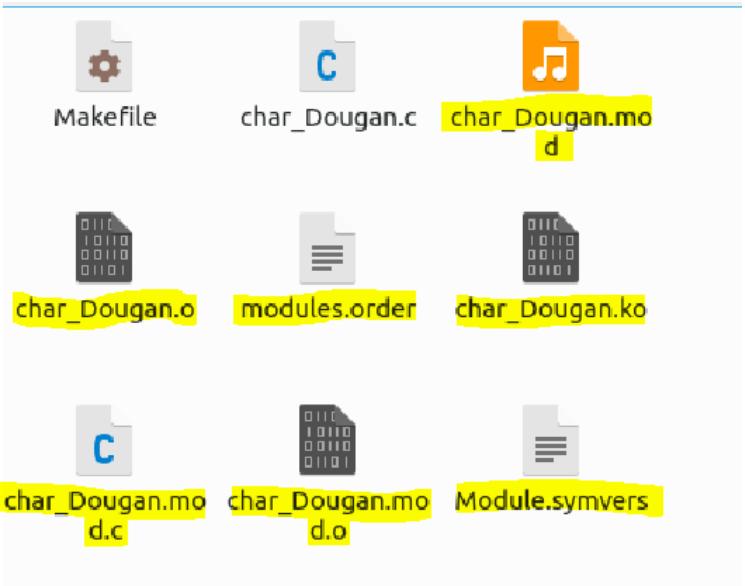
Paso a paso

Nos posicionamos en la carpeta del proyecto y compilamos el archivo Make.

```
alumno@alumno-virtualbox:~/Descargas/v2$ sudo -E make
```

```
alumno@alumno-virtualbox:~/Descargas/v2$ sudo -E make
make -C /lib/modules/5.4.0-48-generic/build M=/home/alumno/Descargas/v2 mo
dules
make[1]: se entra en el directorio '/usr/src/linux-headers-5.4.0-48-generi
c'
  CC [M]  /home/alumno/Descargas/v2/char_Dougan.o
  Building modules, stage 2.
  MODPOST 1 modules
WARNING: modpost: missing MODULE_LICENSE() in /home/alumno/Descargas/v2/ch
ar_Dougan.o
see include/linux/module.h for more information
  CC [M]  /home/alumno/Descargas/v2/char_Dougan.mod.o
  LD [M]  /home/alumno/Descargas/v2/char_Dougan.ko
make[1]: se sale del directorio '/usr/src/linux-headers-5.4.0-48-generic'
```

Luego de ejecutar el Make, dentro de la carpeta veremos generados los siguientes archivos:



Procedemos a cargar el modulo:

```
alumno@alumno-virtualbox:~/Descargas/v2$ sudo insmod char_Dougan.ko
```

Verificamos que el modulo este cargado:

```
alumno@alumno-virtualbox:~/Descargas/v2$ lsmod
Module                  Size  Used by
char_Dougan            16384  0
intel_rapl_msr         20480  0
snd_intel8x0           45056  2
snd_ac97_codec        131072  1 snd_intel8x0
ac97_bus               16384  1 snd_ac97_codec
snd_pcm               106496  2 snd_intel8x0,snd_ac97_codec
snd_seq_midi           20480  0
snd_seq_midi_event     16384  1 snd_seq_midi
snd_rawmidi            36864  1 snd_seq_midi
joydev                 24576  0
intel_rapl_common      24576  1 intel_rapl_msr
snd_seq                69632  2 snd_seq_midi,snd_seq_midi_event
```

Por medio del comando dmesg veremos el *log* del *Kernel*, en donde aparecerá la información del *major number* que fue asignado.

```
[21394.366976] Se genero el  major number 240.
[21394.366978] Ahora se debera crear un dev_file con
[21394.366982] sudo rm /dev/DOUGAN
[21394.366984] sudo mknod /dev/DOUGAN c 240 0
[21394.366985] sudo chmod 666 /dev/DOUGAN
```

Teniendo en cuenta el *major number* que se obtuvo, procedemos a crear el dev_file.

```
alumno@alumno-virtualbox:~$ cd /dev
alumno@alumno-virtualbox:/dev$ sudo mknod DOUGAN c 240 0
[sudo] contraseña para alumno:
alumno@alumno-virtualbox:/dev$ sudo chmod 666 DOUGAN
alumno@alumno-virtualbox:/dev$
```

Para verificar que se encuentre el dev_file se hace un ls, donde deberemos buscar el nombre del dispositivo.

```
alumno@alumno-virtualbox:/dev$ ls
autofs          loop4         tty1   tty40     ttyS12    vboxuser
block           loop5         tty10  tty41     ttyS13    vcs
bsg             loop6         tty11  tty42     ttyS14    vcs1
btrfs-control   loop7         tty12  tty43     ttyS15    vcs2
bus             loop-control  tty13  tty44     ttyS16    vcs3
cdrom           mapper        tty14  tty45     ttyS17    vcs4
char            mcelog        tty15  tty46     ttyS18    vcs5
console         mem           tty16  tty47     ttyS19    vcs6
core            mqueue        tty17  tty48     ttyS2     vcsa
cpu_dma_latency net           tty18  tty49     ttyS20    vcsa1
cuse            null          tty19  tty5      ttyS21    vcsa2
disk            nvram         tty2   tty50     ttyS22    vcsa3
DOUGAN          port          tty20  tty51     ttyS23    vcsa4
dri             ppp           tty21  tty52     ttyS24    vcsa5
dvd             psaux         tty22  tty53     ttyS25    vcsa6
```

Escribimos en el dispositivo.

```
alumno@alumno-virtualbox:/dev$ echo "holaABCD"> DOUGAN
```

Leemos del dispositivo:

```
alumno@alumno-virtualbox:/dev$ cat DOUGAN
mtqfFGHI
```

Como estamos utilizando cifrado con Cesar, en este caso está realizando un movimiento de 5 letras hacia la derecha del abecedario, es por eso que obtuvimos mtqfFGHI.

Eliminamos el dispositivo.

```
alumno@alumno-virtualbox:/dev$ sudo rm DOUGAN
```

Removemos el modulo.

```
alumno@alumno-virtualbox:~/Descargas/v2$ sudo rmmod char_Dougan
```

Aparecerá el mensaje que se ha des-registrado el dispositivo.

```
[21394.366978] Ahora se debera crear un dev_file con
[21394.366982] sudo rm /dev/DOUGAN
[21394.366984] sudo mknod /dev/DOUGAN c 240 0
[21394.366985] sudo chmod 666 /dev/DOUGAN
[21456.613412] El dispositivo desregistrado correctamente
```

Bibliografía

Tldp.org. 2020. [online] Disponible en:
https://tldp.org/LDP/lkmpg/2.6/lkmpg.pdf

Linux Device Drivers, 3rd Edition. (2020) [online] Disponible en:
https://www.oreilly.com/library/view/linux-device-drivers/0596005903/ch03.html

The Linux Kernel API [online] Disponible en:
https://www.fsl.cs.sunysb.edu/kernel-api/