Práctica 4

Entrada/Salida

Objetivo

Aprender los conceptos básicos de la programación de módulos del Kernel Linux a través de la realización de una práctica guiada.

Bibliografía

- P.J. Salzman [et al.], *The Linux Kernel Module Programming Guide*. 2007-05-18 ver 2.6.4. *http://www.tldp.org/guides.html* sección *kernel version 2.6*
- Kernel Documentation *http://www.kernel.org/doc/Documentation/*

4.1. Introducción

En esta práctica se va a seguir el "Linux How-To" de *The Linux Kernel Module Programming Guide* que podemos encontrar en la sección de Bibliografía. Gracias a este manual comprenderemos en qué consiste un módulo del kernel, cómo programarlos y cómo cargarlos dinámicamente para que formen parte del núcleo del Sistema Operativo.

La sección *1. Introduction* del manual ofrece una breve pero explicativa introducción a qué es un módulo y cómo cargarlo en memoria. Debido a que cada distribución tiene sus particularidades, deberemos adaptar los comandos a la versión empleada en el laboratorio, por ejemplo, si consultamos *man modprobe.conf* veremos cómo la versión de Linux Debian "squeeze" no emplea el fichero */etc/modprobe.conf* referido en el apartado *1.2. How Do Modules Get Into The Kernel?*, sino que busca en todos los ficheros *.conf* que se encuentran dentro de */etc/modprobe.d/* y que el uso de un solo fichero será eliminado en futuras versiones del kernel.

Resulta imprescindible saber la versión del kernel que estamos usando, para ello podremos usar el siguiente comando[1]:

```
$ uname -a
Linux debian 2.6.32-5-686 #1 SMP Sun May 6 04:01:19 UTC 2012 i686 GNU/Linux
```

[1]Pese a que se nos insta en la introducción a ejecutar los comandos en una consola fuera de las X, y pese a que es posible cambiar de *tty* en una Máquina Virtual con el comando *sudo chvt N*, se recomienda usar un terminal gráfico normal (*gnome-terminal*).

y ver que estamos usando la versión 686 del 2.6.32-5, por lo que a la hora de consultar las dependencias deberemos fijarnos en el fichero */lib/modules/2.6.32-5-686/modules.dep*.

Cuestión 1: ¿Cuál es la diferencia entre los comandos *modprobe* e *insmod*?

4.2. (Cap 2) Hello World

En esta sección se programarán sencillos módulos del kernel, para lo que deberemos tener en cuenta las siguientes consideraciones:

- El código fuente del kernel se encuentra disponible en */usr/src/linux-source-2.6.32*, donde es posible consultar partes de la documentación, por ejemplo en *Documentation/ kbuild/ modules.txt*.
- Ignorad las indicaciones acerca de consultar los ficheros Makefile y centraos en los ficheros fuente.
- Los ficheros de cabeceras del kernel para la versión instalada en el laboratorio se encuentran en */usr/src/linux-headers-2.6.32-5-686/ include/ linux*.
- La carga y descarga de módulos se tiene que hacer con permisos de administrador. La creación del archivo *Makefile* se puede encontrar en la sección *2.2. Compiling Kernel Modules*, de forma que el proceso de carga del módulo hello-1.c se puede realizar de la siguiente forma:

```
alumno@debian:~$ make
make -C /lib/modules/2.6.32-5-686/build M=/home/alumno modules
make[1]: se ingresa al directorio '/usr/src/linux-headers-2.6.32-5-686'
  CC [M]  /home/alumno/hello-1.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/alumno/hello-1.mod.o
  LD [M]  /home/alumno/hello-1.ko
make[1]: se sale del directorio '/usr/src/linux-headers-2.6.32-5-686'
alumno@debian:~$ su
Contraseña:
root@debian:/home/alumno#
insmod hello-1.ko
```

- No todos los mensajes de *printk* se muestran en */var/log/messages*, lo harán en función de la prioridad y la configuración del kernel. Si queremos ver todos los mensajes podemos consultar:

```
root@debian:/home/alumno# tail /var/log/kern.log
Nov 19 19:24:17 debian kernel: [ 1977.278937] hello_1: module license ...
Nov 19 19:24:17 debian kernel: [ 1977.278942] Disabling lock debugging ...
Nov 19 19:24:17 debian kernel: [ 1977.281473] Hello world 1.
```

  También podemos consultar los mensajes según van apareciendo en el fichero de *log* con el comando *tail -f /var/log/kern.log* o a través de la utilidad gráfica *gnome-system-log*.
- Para ver todos los módulos cargados en el kernel podemos usar la orden *lsmod*. Para borrar un módulo del kernel podemos usar la orden *rmmod*:

```
root@debian:/home/alumno# lsmod
Module                  Size  Used by
hello_1                  532  0
vboxsf                 28713  0
processor              26327  0
...
root@debian:/home/alumno# rmmod hello_1
```

```
root@debian:/home/alumno# tail /var/log/kern.log
Nov 19 19:24:17 debian kernel: [ 1977.278937] hello_1: module license ...
Nov 19 19:24:17 debian kernel: [ 1977.278942] Disabling lock debugging ...
Nov 19 19:24:17 debian kernel: [ 1977.281473] Hello world 1.
Nov 19 19:26:20 debian kernel: [ 2099.461534] Goodbye world 1.
```

Ejercicio 1: Compilar y cargar los códigos correspondientes a los puntos del 2.1 al 2.3 y el 2.6, correspondientes a los códigos *hello-1.c*, *hello-2.c* y *hello-5.c*. El resto de apartados del Capítulo 2 no es necesario probarlos, sólo leerlos y entenderlos.

Cuestión 2: ¿Cuál es la descripción del nivel de urgencia de los mensajes *KERN_EMERG*?

Cuestión 3: ¿Por qué falla la carga e inicialización con parámetros del módulo *hello-5.ko* en el primer ejemplo?:

```
root@debian:/home/alumno# insmod hello-5.ko mystring="bebop" mybyte=255 myintArray=-1
insmod: error inserting 'hello-5.ko': -1 Unknown symbol in module
```

4.3. (Cap 3) Preliminaries

Este capítulo no contiene ejemplos sobre módulos del kernel, pero resulta crucial su entendimiento para comprender las implicaciones que tiene la ejecución en modo kernel. Deberéis leerlo y comprender la diferencia entre espacio de usuario y espacio de kernel en el mapa de memoria. Consideraciones:

- La máquina virtual emula discos duros de tipo *SATA*, no *IDE*, y la nomenclatura para este tipo de discos es *sd** en vez de *hd**, por lo que deberemos usar para mostrar los dispositivos de disco el comando *ls -l /dev/sda[1-5]*

Cuestión 4: ¿Cuáles son los números *Mayor* y *Minor* del disco duro primario, *sda*?

4.4. (Cap 4) Character Device Files

En este capítulo, el tutorial nos muestra un ejemplo de dispositivo de caracteres llamado *chardev*.

Ejercicio 2: Compilar *chardev.c*. Veremos cómo nos da un error:

```
alumno@debian:~$ make
make -C /lib/modules/2.6.32-5-686/build M=/home/alumno modules
make[1]: se ingresa al directorio '/usr/src/linux-headers-2.6.32-5-686'
  CC [M]  /home/alumno/chardev.o
/home/alumno/chardev.c: In function 'cleanup_module':
/home/alumno/chardev.c:72: error: void value not ignored as it ought to be
make[4]: *** [/home/alumno/chardev.o] Error 1
make[3]: *** [_module_/home/alumno] Error 2
make[2]: *** [sub-make] Error 2
make[1]: *** [all] Error 2
make[1]: se sale del directorio '/usr/src/linux-headers-2.6.32-5-686'
make: *** [all] Error 2
```

Podemos ver que hay un error dentro de la función *cleanup_module*, se nos indica que en la línea 72 capturamos un valor de retorno de una función que no lo tiene. Consultando *linux/fs.h*, nos daremos cuenta que para nuestra versión del kernel la función *unregister_chrdev* tiene el siguiente prototipo:

```
static inline void unregister_chrdev(unsigned int major, const char *name)
```

lo que supone que no hay valor de retorno. Deberemos modificar por tanto la función *cleanup_module* para que no espere valor de retorno al dar de baja el dispositivo. Una vez hecho esto, carga el módulo teniendo en cuenta lo siguiente:

- Asegúrate de que el módulo está bien cargado consultando */proc/devices*
- No olvides crear el dispositivo en */dev* con *mknod* y los valores especificados por el módulo una vez cargado
- Lee del dispositivo usando *cat*
- Escribe en el dispositivo mediante *echo* ¡Ojo con los permisos!

Cuestión 4: ¿Cuáles son los números *Mayor* y *Minor* del dispositivo *chardev*?