UNIVERSIDAD
DE GRANADA

MEMORIA Y DOCUMENTACIÓN

SEMINARIO LKM

RAFAEL DELGADO GARCÍA-VALDECASAS

PABLO RIENDA SÁNCHEZ

GRADO EN INGENIERÍA INFORMÁTICA

Curso 2023-2024

1. Introducción

En este seminario aprenderemos sobre cómo funciona el sistema de módulos cargables del kernel de Linux. Nuestra tarea será realizar un módulo sencillo.

2. Módulo sencillo

Primero, comprobamos qué cabeceras de Linux necesitamos:

```
rafadgvc@ubuntuUni:~/pdih$ uname -r
6.5.0-27-generic
```

A continuación, comprobamos que tenemos instaladas las cabeceras correspondientes a nuestro kernel:

```
rafadgvc@ubuntuUni:~/pdih$ sudo apt-get install linux-headers-6.5.0.27-generic
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Nota, seleccionando «linux-headers-6.5.0-27-generic» para la expresión regular «linux-headers-6.5.0.27-generic»
linux-headers-6.5.0-27-generic ya está en su versión más reciente (6.5.0-27.28~22.04.1).
fijado linux-headers-6.5.0-27-generic como instalado manualmente.
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 183 no actualizados.
```

Después, partiendo del material proporcionado, creamos una versión modificada del hello.c y nos vamos al directorio en el que se encuentra junto con el Makefile.

Abrir

hello.c
~/pdih/modulo

```
 6  * @brief  An introductory "Hello World!" loadable kernel module (LKM) that can display a message
 7  * in the /var/log/kern.log file when the module is loaded and removed. The module can accept an
 8  * argument when it is loaded -- the name, which appears in the kernel log files.
 9  * @see http://www.derekmolloy.ie/ for a full description and follow-up descriptions.
10 */
11
12 #include <linux/init.h>             // Macros used to mark up functions e.g., __init __exit
13 #include <linux/module.h>           // Core header for loading LKMs into the kernel
14 #include <linux/kernel.h>           // Contains types, macros, functions for the kernel
15
16 MODULE_LICENSE("GPL");              ///< The license type -- this affects runtime behavior
17 MODULE_AUTHOR("Derek Molloy");      ///< The author -- visible when you use modinfo
18 MODULE_DESCRIPTION("A simple Linux driver for the BBB.");  ///< The description -- see modinfo
19 MODULE_VERSION("0.1");              ///< The version of the module
20
21 static char *name = "world";        ///< An example LKM argument -- default value is "world"
22 module_param(name, charp, S_IRUGO); ///< Param desc. charp = char ptr, S_IRUGO can be read/not changed
23 MODULE_PARM_DESC(name, "The name to display in /var/log/kern.log");  ///< parameter description
24
25 /** @brief The LKM initialization function
26  *  The static keyword restricts the visibility of the function to within this C file. The __init
27  *  macro means that for a built-in driver (not a LKM) the function is only used at initialization
28  *  time and that it can be discarded and its memory freed up after that point.
29  *  @return returns 0 if successful
30  */
31 static int __init helloBBB_init(void){
32    printk(KERN_INFO "EBB: Hola %s desde el LKM de BBB!\n", name);
33    return 0;
34 }
35
36 /** @brief The LKM cleanup function
37  *  Similar to the initialization function, it is static. The __exit macro notifies that if this
38  *  code is used for a built-in driver (not a LKM) that this function is not required.
39  */
40 static void __exit helloBBB_exit(void){
41    printk(KERN_INFO "EBB: Adiós %s desde el LKM de BBB!\n", name);
42 }
43
44 /** @brief A module must use the module_init() module_exit() macros from linux/init.h, which
45  *  identify the initialization function at insertion time and the cleanup function (as
46  *  listed above)
47  */
48 module_init(helloBBB_init);
49 module_exit(helloBBB_exit);
```

Hemos cambiado el mensaje que se devuelve de “hello name from the BBB Linux” a “Hola name desde LKM de BBB”. También hemos cambiado el mensaje de adiós a “Adiós name desde el LKM de BBB”.

A continuación hacemos make para compilar el archivo hello.c:

```
rafadgvc@ubuntuUni:~/pdih/modulo$ make
make -C /lib/modules/6.5.0-27-generic/build/ M=/home/rafadgvc/pdih/modulo modules
make[1]: se entra en el directorio '/usr/src/linux-headers-6.5.0-27-generic'
warning: the compiler differs from the one used to build the kernel
  The kernel was built by: x86_64-linux-gnu-gcc-12 (Ubuntu 12.3.0-1ubuntu1~22.04) 12.3.0
  You are using:           gcc-12 (Ubuntu 12.3.0-1ubuntu1~22.04) 12.3.0
  CC [M]  /home/rafadgvc/pdih/modulo/hello.o
  MODPOST /home/rafadgvc/pdih/modulo/Module.symvers
  CC [M]  /home/rafadgvc/pdih/modulo/hello.mod.o
  LD [M]  /home/rafadgvc/pdih/modulo/hello.ko
  BTF [M] /home/rafadgvc/pdih/modulo/hello.ko
Skipping BTF generation for /home/rafadgvc/pdih/modulo/hello.ko due to unavailability of vmlinux
make[1]: se sale del directorio '/usr/src/linux-headers-6.5.0-27-generic'
```

Se genera el archivo hello.ko, que es el módulo LKM compilado.

Ahora cargamos el módulo hello.ko con el comando sudo insmod hello.ko. Para ver si el módulo se ha cargado ejecutamos el comando lsmod | grep hello.

```
rafadgvc@ubuntuUni:~/pdih/modulo$ sudo insmod hello.ko
rafadgvc@ubuntuUni:~/pdih/modulo$ lsmod | grep hello
hello                  12288  0
```

Podemos ver información del módulo con el comando modinfo hello.ko.

```
rafadgvc@ubuntuUni:~/pdih/modulo$ modinfo hello.ko
filename:       /home/rafadgvc/pdih/modulo/hello.ko
version:        0.1
description:    A simple Linux driver for the BBB.
author:         Derek Molloy
license:        GPL
srcversion:     B99C704DCD1B50A14C269CF
depends:
retpoline:      Y
name:           hello
vermagic:       6.5.0-27-generic SMP preempt mod_unload modversions
parm:           name:The name to display in /var/log/kern.log (charp)
```

Ejecutamos el módulo con el comando rmmod hello.ko:

```
rafadgvc@ubuntuUni:~/pdih/modulo$ sudo rmmod hello.ko
```

Para ver el resultado del módulo tenemos que ver el log del kernel.

```
rafadgvc@ubuntuUni:/var/log$ tail -f kern.log
Jun  3 10:13:55 ubuntuUni kernel: [  507.934753] audit: type=1400 audit(1717402435.330:72): apparmor="STATUS" operation="profile_replace" profile="unc
onfined" name="snap.firefox.geckodriver" pid=5591 comm="apparmor_parser"
Jun  3 10:13:55 ubuntuUni kernel: [  508.156022] audit: type=1400 audit(1717402435.558:73): apparmor="STATUS" operation="profile_replace" profile="unc
onfined" name="snap.firefox.hook.configure" pid=5592 comm="apparmor_parser"
Jun  3 10:13:55 ubuntuUni kernel: [  508.305718] audit: type=1400 audit(1717402435.702:74): apparmor="STATUS" operation="profile_replace" profile="unc
onfined" name="snap.firefox.hook.connect-plug-host-hunspell" pid=5593 comm="apparmor_parser"
Jun  3 10:13:55 ubuntuUni kernel: [  508.418596] audit: type=1400 audit(1717402435.822:75): apparmor="STATUS" operation="profile_replace" profile="unc
onfined" name="snap.firefox.hook.disconnect-plug-host-hunspell" pid=5594 comm="apparmor_parser"
Jun  3 10:13:55 ubuntuUni kernel: [  508.547542] audit: type=1400 audit(1717402435.950:76): apparmor="STATUS" operation="profile_replace" profile="unc
onfined" name="snap.firefox.hook.post-refresh" pid=5595 comm="apparmor_parser"
Jun  3 10:17:45 ubuntuUni kernel: [  737.770953] hrtimer: interrupt took 9542117 ns
Jun  3 10:40:51 ubuntuUni kernel: [ 1249.432900] hello: loading out-of-tree module taints kernel.
Jun  3 10:40:51 ubuntuUni kernel: [ 1249.432944] hello: module verification failed: signature and/or required key missing - tainting kernel
Jun  3 10:40:51 ubuntuUni kernel: [ 1249.441009] EBB: Hola world desde el LKM de BBB!
Jun  3 10:41:25 ubuntuUni kernel: [ 1284.062789] EBB: Adiós world desde el LKM de BBB!
```