Device drivers- Linux Kernel Module

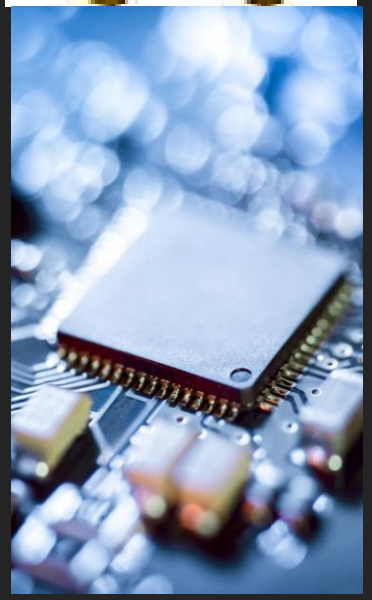
Dr. Sukesh Rao M

Associate Professor

E&C dept.

NMAMIT, Nitte

Email: sukesh@nitte.edu.in



Device Drivers

- The role of a driver is to provide mechanisms which allows normal user to access protected parts of its system, in particular ports, registers and memory addresses normally managed by the operating system
- Users can add or remove functionalities to the kernel while the system is running
- These “programs” that can be added to the kernel at runtime are called “Linux Kernel Module” and built into individual files with .ko (Kernel object) extension

Types of Device Drivers

- **Character Type**
- “Char” devices are devices that can be accessed as a stream of bytes (like a file)
- **Block Type**
- “Block” devices are accessed by filesystem nodes (example: disks).
- **Network Type**
- “Network” interfaces are able to exchange data with other hosts
- Linux driver modules can be found in: **/lib/modules/<version>/kernel/drivers/**
- where <version> would be the output of the command "uname -r" on the system

Command	Function
`lsmod`	List the currently loaded modules
`insmod module`	Load the module specified by `module`
`modprobe module`	Load the module along with its dependencies
`rmmod module`	Remove/Unload the module

- In order to write, modify, compile and upload a device driver, the user needs temporarily superuser (root) permissions
- Module management can be done with four basic shell commands:

Structure of LKM

- A device driver contains at least two functions:
- A function for the module initialization (executed when the module is loaded with the command "insmod")
- A function to exit the module (executed when the module is removed with the command "rmmod")
- These two are like normal functions in the driver, except that these are specified as the init and exit functions, respectively, by the macros
- **module_init()** and **module_exit()**
- which are defined in the kernel header **module.h**

simple kernel module in C

```
#include <linux/module.h>
#include <linux/version.h>
#include <linux/kernel.h>

static int __init init_mod(void) /* Constructor */
{
   printk(KERN_INFO "Module1 started...\n");
   return 0;
}

static void __exit end_mod(void) /* Destructor */
{
   printk(KERN_INFO "Module1 ended...\n");
}

module_init(init_mod);
module_exit(end_mod);
```

Procedure to compile LKM

- The LKM source code should be compiled using linux kernel build directory
- A Makefile configured or Make command is used to compile the source code with the required linux kernel version
- After the compilation .ko file is generated from the linux kernel build directory
- “insmod” command is used to load the LKM or .ko file

Makefile

- ***obj-m := source_file.o***
- The above statement indicates, source_file.o should be linked to get object file of LKM (.ko)

- ***make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules***
- This command compiles the object file of .o with kbuild directory stored in
  - ***/lib/modules/$(shell uname -r)/build*** → current kernel
  - ***or***
  - ***/lib/modules/4.15.0-118-generic/build*** → Kernel version
- ***M=$(PWD) modules***
- ***M*** holds the location of current kernel module path and ***modules*** tells the kbuild to generate module

Typical Makefile

Makefile (~/linux_kernel) - gedit

Open

```
obj-m := hello.o
obj-m += hello1.o
obj-m += hello2.o

all:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

- It is also possible to declare just
- ***obj-m := source_file.o*** in the ***Makefile*** and run the compilation command on the terminal as
- ***make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules***

Development of simple LKM

LKM Source file

hello.c (~/linux_kernel) - gedit

Open

```
#include <linux/init.h>
#include <linux/module.h>
int hello_init(void)
{
        printk(KERN_ALERT "I am inside kernel \n");
        return 0;
}

void hello_exit(void)
{
        printk(KERN_ALERT " Leaving the kernel, bye \n" );
}
module_init(hello_init);
module_exit(hello_exit);
```

Run make Command

Makefile (~/linux_kernel) - gedit

Open

```
obj-m := hello.o
```

Makefile

```
ece@ece-TM4750:~/linux_kernel$ make -C /lib/modules/4.15.0-122-gene
ric/build M=$PWD modules
make: Entering directory '/usr/src/linux-headers-4.15.0-122-generic
'
  CC [M]  /home/ece/linux_kernel/hello.o
  Building modules, stage 2.
  MODPOST 1 modules
WARNING: modpost: missing MODULE_LICENSE() in /home/ece/linux_kerne
l/hello.o
see include/linux/module.h for more information
  CC      /home/ece/linux_kernel/hello.mod.o
  LD [M]  /home/ece/linux_kernel/hello.ko
make: Leaving directory '/usr/src/linux-headers-4.15.0-122-generic'
```

Using make all

Makefile (~/linux_kernel) - gedit

Open

```
obj-m := hello.o

all:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
	make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

Makefile

```
ece@ece-TM4750:~/linux_kernel$ make all
make -C /lib/modules/4.15.0-122-generic/build M=/home/ece/linux_ker
nel modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-122-gene
ric'
  Building modules, stage 2.
  MODPOST 1 modules
WARNING: modpost: missing MODULE_LICENSE() in /home/ece/linux_kerne
l/hello.o
see include/linux/module.h for more information
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-122-gener
ic'
```

lsmod

- List the kernel modules currently being loaded

```
ece@ece-TM4750:~/linux_kernel$ lsmod
Module                  Size  Used by
bnep                   20480  2
vboxnetadp             28672  0
vboxnetflt             28672  0
vboxdrv               483328  2 vboxnetadp,vboxnetflt
snd_hda_codec_hdmi     49152  1
snd_hda_codec_realtek   106496  1
ath3k                  20480  0
btusb                  45056  0
intel_rapl             20480  0
snd_hda_codec_generic    73728  1 snd_hda_codec_realtek
btrtl                  16384  1 btusb
btbcm                  16384  1 btusb
snd_hda_intel          45056  3
x86_pkg_temp_thermal    16384  0
intel_powerclamp       16384  0
```

insmod

- Insert the LKM (.ko) file with sudo permission using **insmod**

```
ece@ece-TM4750:~/linux_kernel$ sudo insmod hello.ko
[sudo] password for ece:
ece@ece-TM4750:~/linux_kernel$
```

- Check the loaded LKM (.ko) using **lsmod**

```
ece@ece-TM4750:~/linux_kernel$ lsmod
Module                  Size  Used by
hello                  16384  0
bnep                   20480  2
vboxnetadp             28672  0
vboxnetflt             28672  0
```

hello.ko